Fundamentación de la electrodinámica

Segundo mini-proyecto

Propagación de ondas en espacio libre

Profesor Julio César Gutiérrez Vega

Arif Morán Velázquez — A01234442

Alberto Anaya Velasco — A01252512

Ing. Física Industrial

Monterrey, Nuevo León

18 de junio de 2023

I. Objetivo y antecedentes

El objetivo de esta actividad es simular la propagación libre de ondas electromagnéticas. Con la idea de comparar nuestras rutinas contra resultados teóricos conocidos, se toma como ejemplo el clásico experimento de Young unidimensional. Una vez que sea seguro que las rutinas funcionan adecuadamente, se propagarán frentes de luz más complicados y que no tengan solución analítica exacta.

Se considera el clásico experimento de Young de la doble rendija mostrado en la Fig. 1(a). El ancho de cada rendija es *b* y están separadas por una distancia *h*. Si una onda plana monocromática de longitud de onda λ incide frontalmente sobre la pantalla el patrón de difracción de Fraunhofer para el caso unidimensional tiene una distribución de intensidad proporcional a

$$I \propto \left(\frac{\sin\beta}{\beta}\right)^2 \cos^2\gamma, \tag{1}$$

donde $\beta = \frac{1}{2}kb\sin\theta$, y, finalmente, $k = 2\pi/\lambda$ es la constante de propagación. Los detalles de la derivación de la Ec. (1) pueden consultarse en muchas referencias, p. ej. [1, Sect. 10.2.2].

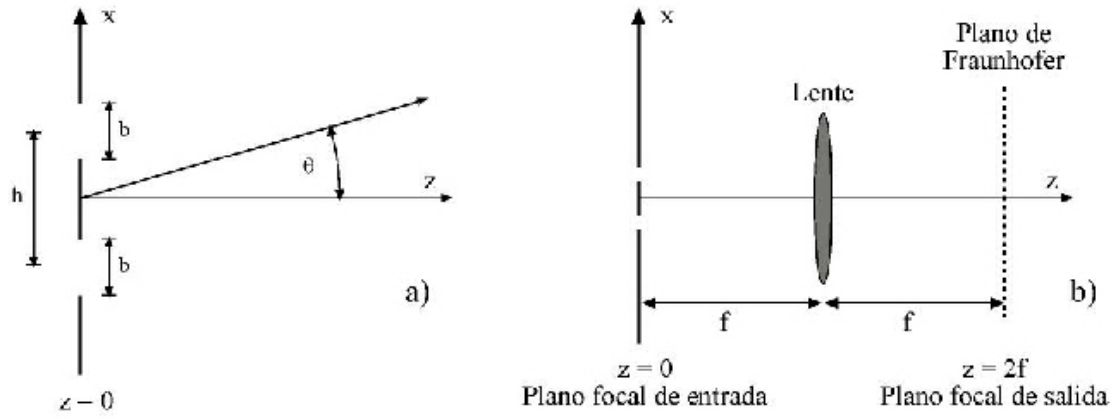


Figura 1. Geometría del experimento de Young

El factor $\left(\frac{\sin\beta}{\beta}\right)^2$ en la Ec. (1) corresponde al patrón de intensidad para una rendija única. Aquí este factor contribuye como una envolvente para las franjas de interferencia de Young dadas por el término $\cos^2\gamma$. Franjas brillantes ocurren para $\gamma = 0, \pm\pi, \pm 2\pi, \ldots$. La separación angular entre franjas está dada por $\Delta\gamma = \pi$, o aproximadamente en la regió paraxial, en términos del ángulo θ

$$\Delta\theta \approx \frac{2\pi}{kh} = \frac{\lambda}{h}. \tag{2}$$

La propagación libre de la onda electromagnética se calcula solucionando la ecuación escalar de onda que en el régimen paraxial toma la forma

$$\frac{\partial^2 U}{\partial x^2} + i2k\frac{\partial U}{\partial z} = 0, \tag{3}$$

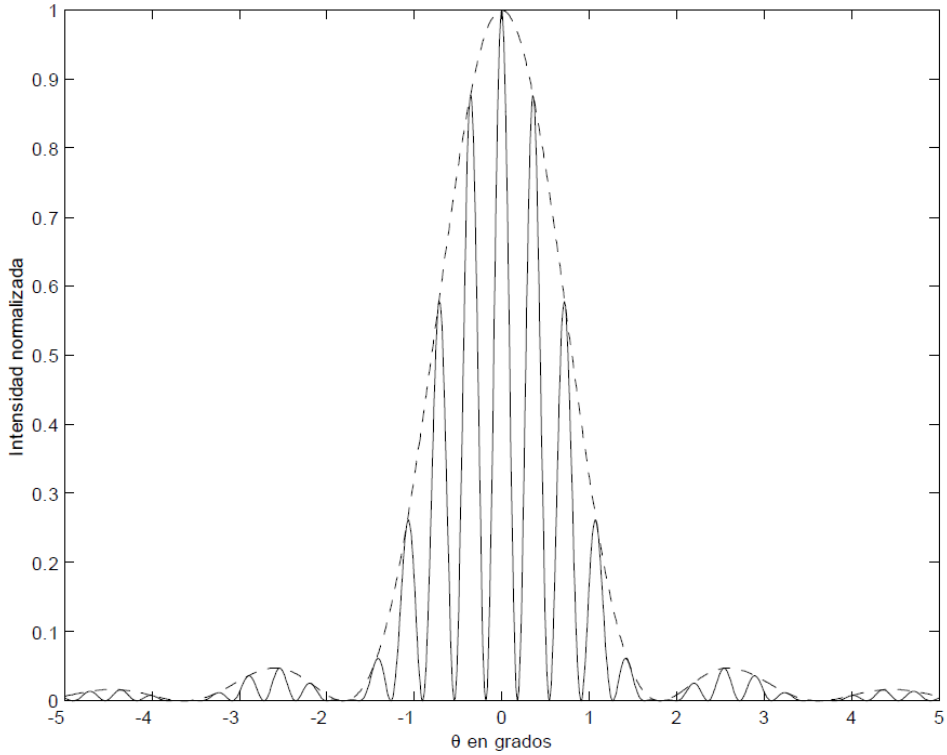


Figura 2. Gráfica de la intensidad relativa $I(\theta)$ dada por la Ec. (1) correspondiente a un experimento de Young. La línea punteada es la envolvente $sinc$ de la variación cosenoidal. Para esta gráfica $\lambda =$ 632.8 nm, $h = 1$ mm y $b = 0.2$ mm.

donde $U(x, z)$ es la amplitud del campo eléctrico de la onda, i es el número imaginario $\sqrt{-1}$. Observa que la ecuación paraxial de onda tiene la estructura de una ecuación de difusión unidimensional con la variable z en vez de la variable t.

II. Método de Crank-Nicolson.[1]

De la ecuación (3),

$$\frac{\partial^2 U}{\partial x^2} + i2k\frac{\partial U}{\partial z} = 0,$$

$$\frac{\partial^2 U}{\partial x^2} = -i2k\frac{\partial U}{\partial z}.$$

Utilizando la forma

$$\kappa\frac{\partial^2 U}{\partial x^2} = \frac{\partial U}{\partial t},$$

donde

$$\kappa = -\frac{1}{i2k},$$

se aplican los métodos FTCS (tempo hacia delante, centrado en el espacio), explícito, y BTCS (tiempo hacia atrás, centrado en el espacio), implícito, para obtener por diferencias finitas el método de Crank-Nicolson:

$$\kappa\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{2\Delta x^2} + \kappa\frac{U_{i+1,j+1} - 2U_{i,j+1} + U_{i-1,j+1}}{2\Delta x^2} + i2k\frac{U_{i,j+1} - U_{i,j}}{\Delta z} = 0,$$

$$\frac{\kappa\Delta z}{\Delta x^2}\left(U_{i+1,j} - 2U_{i,j} + U_{i-1,j} + U_{i+1,j+1} - 2U_{i,j+1} + U_{i-1,j+1}\right) = 2\left(U_{i,j+1} - U_{i,j}\right),$$

donde

$$\alpha \equiv \frac{\kappa\Delta z}{\Delta x^2}.$$

Reorganizando de la forma $U_{j+1} = U_j$:

$$-\alpha U_{i-1,\ n+1} + 2(1+\alpha)U_{i,\ n+1} - \alpha U_{i+1,\ n+1} = \alpha U_{i-1,n} + 2(1-\alpha)U_{i,n} + \alpha U_{i+1,n}\ .$$

En formato matricial, $\boldsymbol{C}\boldsymbol{f}_{n+1} = \boldsymbol{D}\boldsymbol{f}_n$:

$$\begin{bmatrix} 2(1+\alpha) & -\alpha & 0 & 0 \\ -\alpha & 2(1+\alpha) & -\alpha & 0 \\ 0 & -\alpha & 2(1+\alpha) & -\alpha \\ 0 & 0 & -\alpha & 2(1+\alpha) \end{bmatrix}\boldsymbol{f}_{n+1} = \begin{bmatrix} 2(1-\alpha) & \alpha & 0 & 0 \\ \alpha & 2(1-\alpha) & \alpha & 0 \\ 0 & \alpha & 2(1-\alpha) & \alpha \\ 0 & 0 & \alpha & 2(1-\alpha) \end{bmatrix}\boldsymbol{f}_{\mathrm{n}},$$

donde

$$\boldsymbol{f}_n = \begin{bmatrix} f_{1,n} \\ f_{2,n} \\ \vdots \\ f_{K,n} \end{bmatrix}.$$

Resolviendo para $\boldsymbol{f}_{n+1}$,

$$\boldsymbol{C}\boldsymbol{f}_{n+1} = \boldsymbol{D}\boldsymbol{f}_n,$$

$$\boldsymbol{P} \equiv \boldsymbol{C}^{-1}\boldsymbol{D},$$

y, finalmente se obtiene:

$$\boldsymbol{f}_{j+1} = \boldsymbol{P}\boldsymbol{f}_j \, .$$

III. Validación de los pulsos

Primeramente, para comenzar con la propagación de ondas, se graficó la propagación de dos ondas gaussianas, sin simular las rendijas aún, para comprobar que la rutina de programación funciona correctamente. Ingresando la condición inicial,

$$U_{i,1} = e^{\frac{-\left(x-\frac{h}{2}\right)^2}{k_1}} + e^{\frac{-\left(x+\frac{h}{2}\right)^2}{k_1}} ; k_1 \neq k;\ k_1 = 1 \times 10^{-4}\ mm,$$

donde $h = 4\ mm$, se obtiene la propagación de la **Figura 3**.

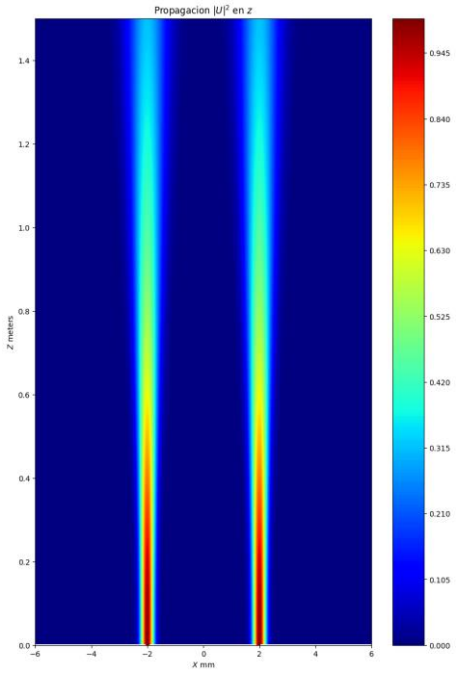


Figura 3. Propagación de dos pulsos gaussianos sin rendija.

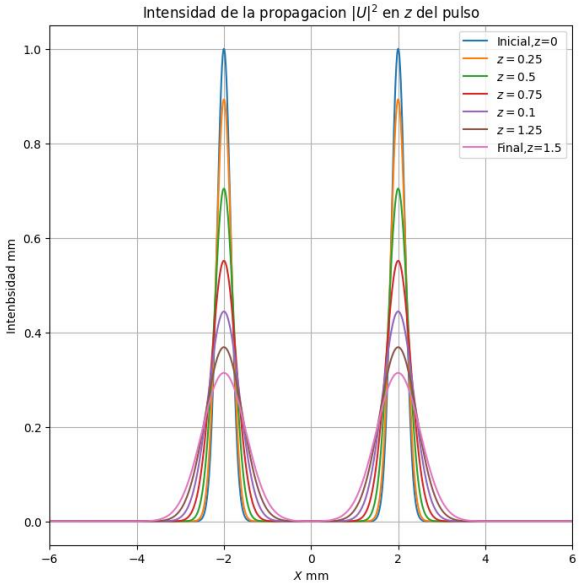


Figura 4. Intensidad de propagación de los pulsos en diferentes valores de z.

Se repite el procedimiento, multiplicando el campo por una transmitancia para probar el funcionamiento de la lente en $f = 0.75\ m$:

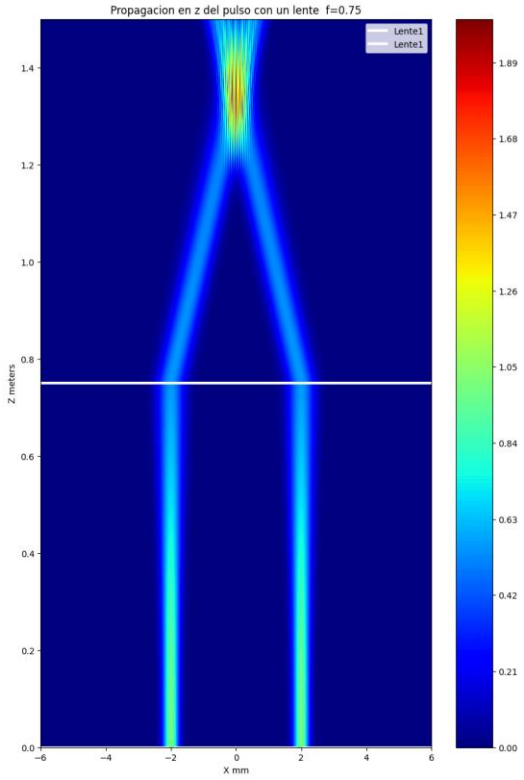


Figura 5. Validación del funcionamiento de la lente al propagar dos pulsos gaussianos.

En un segundo caso, se probó la rutina en una sola onda gaussiana, simulándola también con una lente.

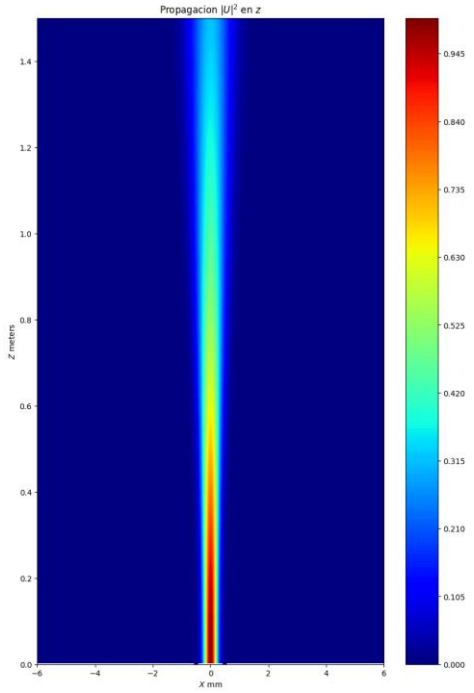


Figura 6. Propagación de un pulso solitario sin rendija.

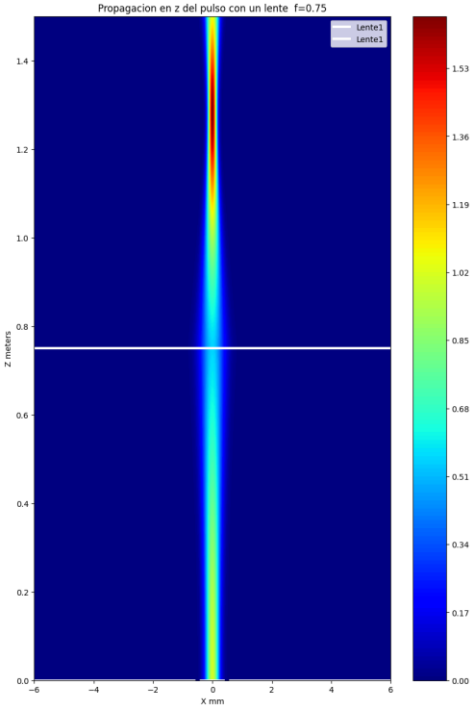


Figura 7. Propagación de un pulso solitario con lente.

En las simulaciones se puede observar el comportamiento conocido de las ondas o pulsos gaussianos, en la cuales se puede ver cómo a lo largo del eje *z* el pulso se va atenuando. Una vez que se comprobó el comportamiento de los códigos, tanto en la propagación como el enfoque de la lente, se procedió a propagar dos secantes hiperbólicas centradas en las rendijas con un ancho cercano a las rendijas.

IV. Simulación del experimento de Young

Se continúa utilizando el método de Crank-Nicolson para resolver la ecuación escalar de onda en el régimen paraxial y simular la propagación de la luz en un experimento de Young. Se toma como campo inicial en $z = 0$ la luz que sale justo después de la pantalla con la doble rendija. También se consideran los siguientes parámetros para la simulación:

- $z \in [0,1.5]\, m$ y $x \in [-6,6]\, mm$, el rango de propagación y transversal, respectivamente;
- $\lambda = 632.8\, nm = 632.8\, e^{-9} m$, que corresponde a la longitud de onda de la luz de un láser de He-Ne;
- $n_z = 200$, el número de discretizaciones longitudinales;
- $n_x = 511$, el número de discretizaciones transversales;
- $h = 1\, mm$ y $b = 0.2\, mm$ para la rendija, según la **Figura 1**.

Nuevamente, se define el vector inicial que contiene la onda, pero se reemplaza con ceros aquellos lugares en los que se encuentra la rendija bloqueando el pulso, y por lo tanto solo algunos de los datos entran a la rendija.

En esta ocasión una onda inicial que está dada por:

$$U_{i,1} = \operatorname{sech}\left(\frac{x - \frac{h}{2}}{k_1}\right) + \operatorname{sech}\left(\frac{x + \frac{h}{2}}{k_1}\right),$$

con las condiciones iniciales descritas de rendija y donde $k_1 \neq k; k_1 = 1 \times 10^{-4}\, m$.

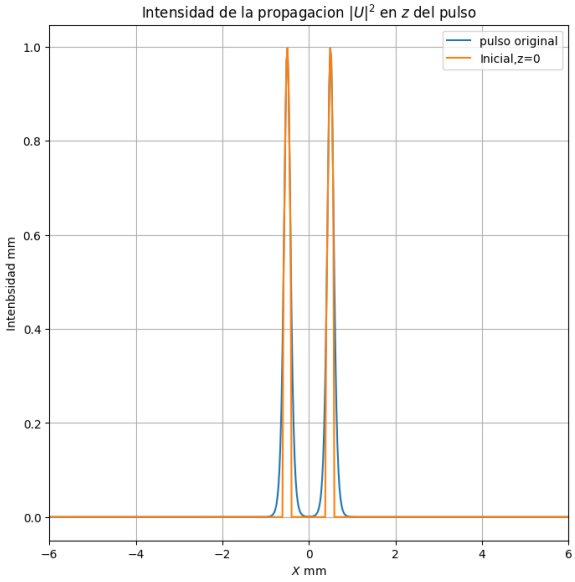


Figura 8. Pulso original y pulso simulando la rendija en $z = 0$. Se observa la delimitación de las rendijas, en las cuales primero se define la función y después se hacen ceros aquellos elementos que no corresponde en la rendija. De esta forma solo sobrevive la función en aquellos puntos dentro de las rendijas.

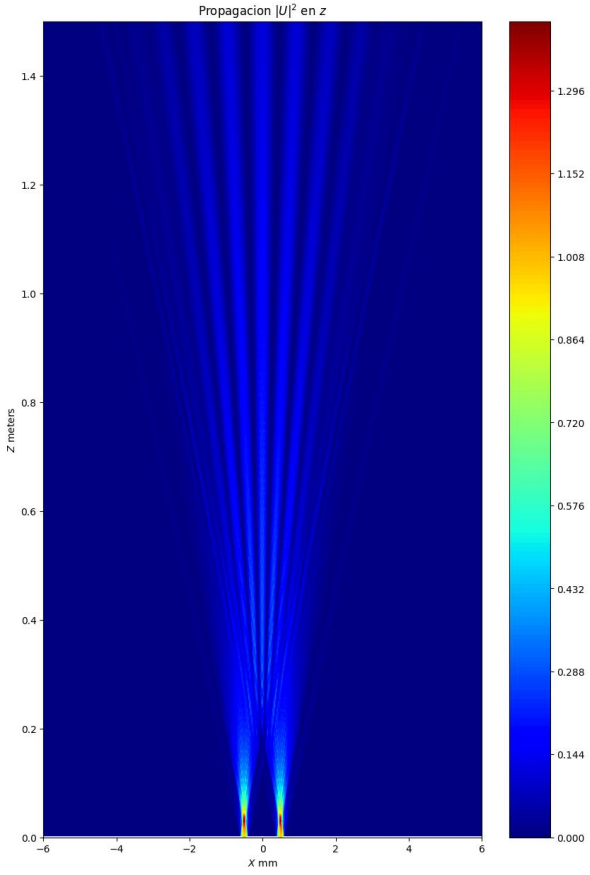


Figura 9. Propagación en z sin lente con rendijas

V. Comprobaciones

Ahora se calcula la energía del haz usando

$$Energía\ (z) \propto \int U^2\, dx \approx \sum_j \left[abs\left(U_j\right)\right]^2 \Delta x,$$

donde j es el contador de puntos transversales. Esto se realiza con el motivo de monitorear una cantidad que se conserva y así revisar la estabilidad y correcto funcionamiento de la simulación.

Ya con el cálculo de la energía, se calcula el error relativo porcentual utilizando

$$ERP\ (z) = 100 \cdot \left|\frac{E_{inicial} - Energía\ (z)}{E_{inicial}}\right|.$$

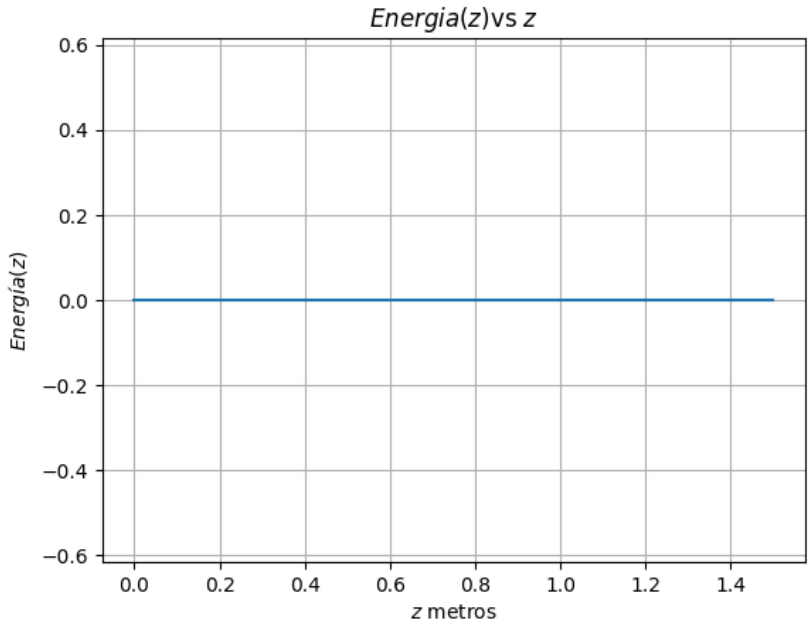


Figura 10. Cálculo de energía contra distancia (z).

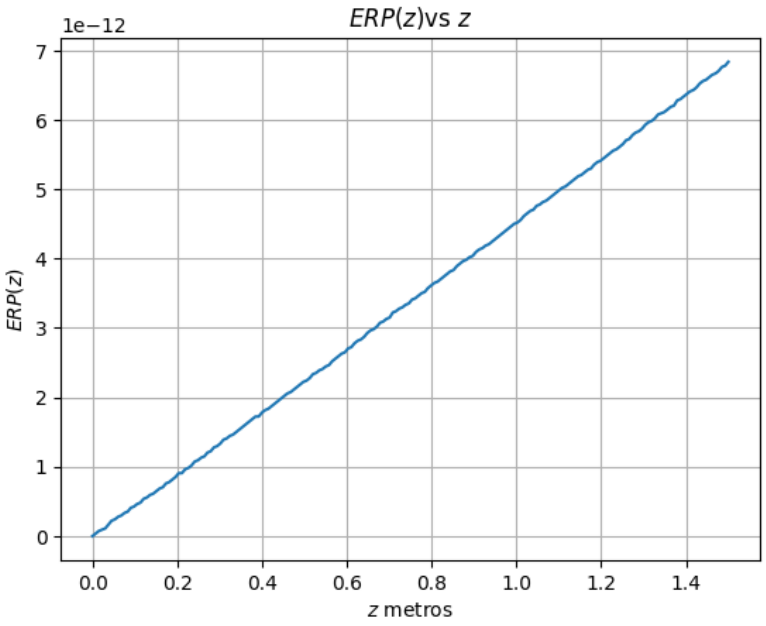


Figura 11. Cálculo del error $ERP(z)$ a lo largo de z.

Con esto se confirma que $ERP(z) << 0.001$, por lo que la trayectoria simulada se puede considerar correcta. Empalmando las gráficas unidimensionales de la intensidad del campo para diferentes valores de z, se puede observar que el comportamiento del pulso en z es el esperado:

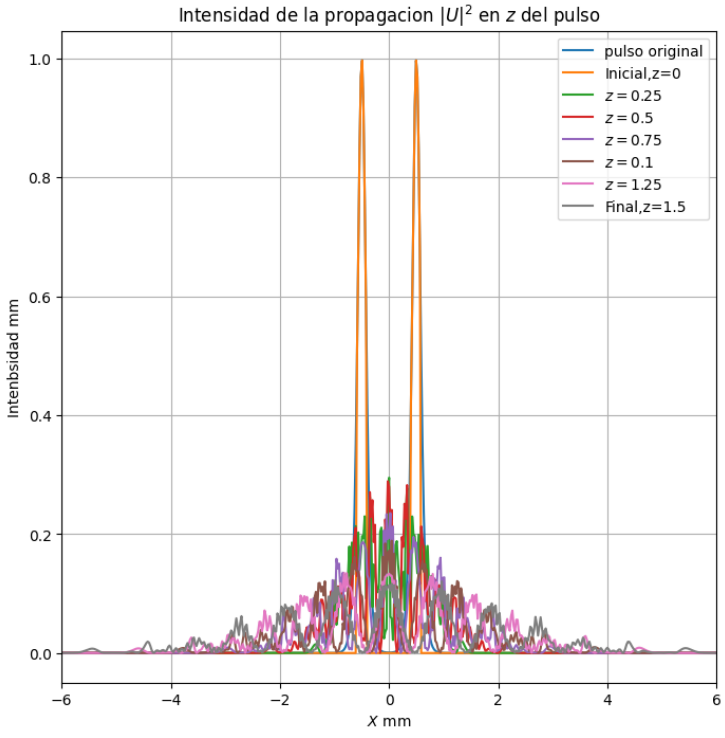


Figura 12. Propagacion sin lente en diferentes z

Además, se corroboran los resultados numéricos obtenidos anteriormente por medio de las predicciones teóricas dadas por las Ecs. (1) y (2).

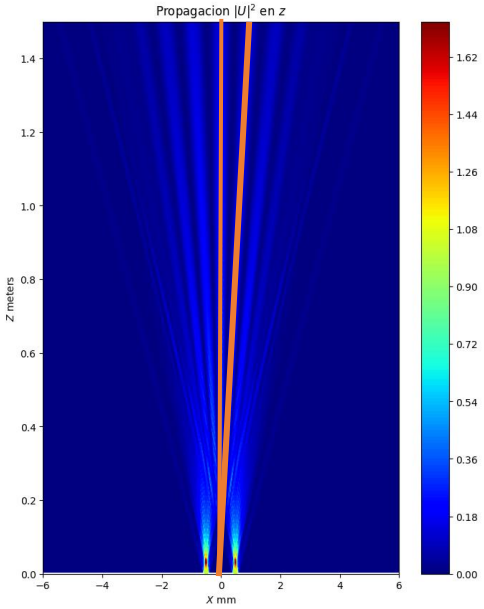


Figura 13.13

Para demostrar la Ecuación (2), utilizamos el triángulo en la **Figura 13** para obtener el espacio que existe entre franjas Δx. Para obtener la distancia entre los picos de la onda se utilizó la primer y segunda derivada: primero se suavizó la función de la onda final, y en la primera derivada se obtuvieron aquellos puntos que cruzaban el cero, mientras que en la segunda derivada se obtenían los elementos menores a 0, indicando un máximo. Si bien filtraba una gran cantidad de elementos, aún existía un error enorme en la cantidad picos. Para refinar este resultado, se usó un método de la librería de *Scipy* que permitió encontrar los picos; sin embargo, también generaba más picos de los que debería. Finamente se compararon y tomaron los puntos presentes tanto en las derivadas como en la función de *Scipy*. Esto permitió encontrar los puntos en x de los puntos de los picos de la función, así como se ve en la **Figura 14** y de ahí el Δx.

A partir de allí, simplemente se despeja el ángulo. El resultado arrojó un valor muy cercano al esperado teórico:

$$\Delta\theta_{teorico} \approx \frac{\lambda}{h} = 6.328 * 10^{-4}\ rads,$$

$$\overline{\Delta\theta_{numerico}} = \tan^{-1}\left(\frac{\overline{\Delta x}}{z}\right) = 6.2 * 10^{-4}\ rads.$$

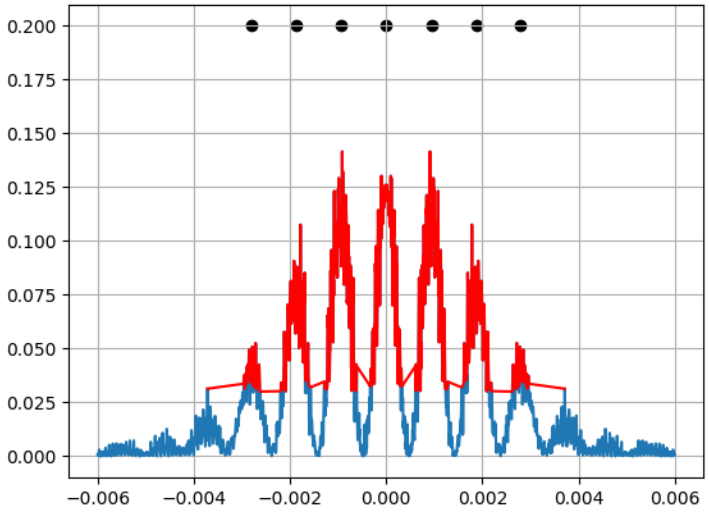


Figura 14. Distancia entre máximos de la onda final (mm).

Finalmente, se tiene un error en la comprobación de la Ecuación (2) de

$$Error = \left|\frac{\Delta\theta_{teorico} - \Delta\theta_{numerico}}{\Delta\theta_{teorico}}\right| * 100\% = 2.022\% \, .$$

Ahora, para corroborar la Ecuación (1), se comparan ambas gráficas de la intensidad. De la Ec. (1), se tiene que

$$I_{teorica} \propto \left(\frac{sin\beta}{\beta}\right)^2 cos^2\gamma,$$

donde, $\beta = \frac{1}{2} kb \, sin\theta$ y $\gamma = \frac{1}{2} kh \, sin(\theta)$.

Para comparar la función de nuestra amplitud de la función final se calculó los ángulos en grados a partir de los puntos en x y la distancia z. Por otro lado, el vector con la amplitud se normalizo y se escaló. Al graficar los resultados se obtuvo la figura 15. Al analizar esta figura vemos que se ajusta de una manera muy apegada a la intensidad dada por la ecuación (1).

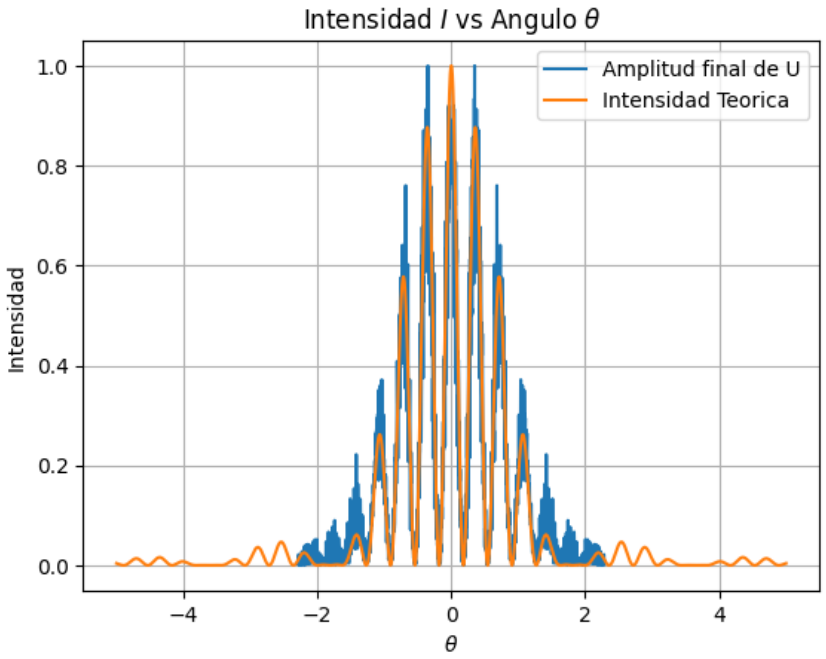


Figura 15. Intensidad teórica y Amplitud final vs Angulo en grados.

Al graficar la amplitud al final del recorrido (en $z = 1.5$), se tiene un comportamiento idéntico, solo diferente en proporción, a la intensidad teórica dada por la *ecuación (1).*

VI. Simulación con una lente convergente

En este apartado se revisa la propagación en el caso de colocar una lente convergente a la mitad de la propagación, como en la Fig. 1(b). Para esto propusimos el campo como se hizo en el Problema 1, con la lente. Para este caso, cuando la propagación alcanza el punto f, se multiplica la propagación por la transmitancia de una lente convergente,

$$T_L(x) = \exp\left(-i\frac{k}{2f}x^2\right)$$

donde f es la distancia focal de la lente. En este caso se selecciona $f = 0.75$ m, de tal forma que la pantalla quede en el plano focal de la lente. Luego, se continúa propagando libremente hasta $z = 1.5\,m$, de tal forma que la última iteración muestra el segundo plano focal, lo que equivale al plano de Fraunhofer [1].

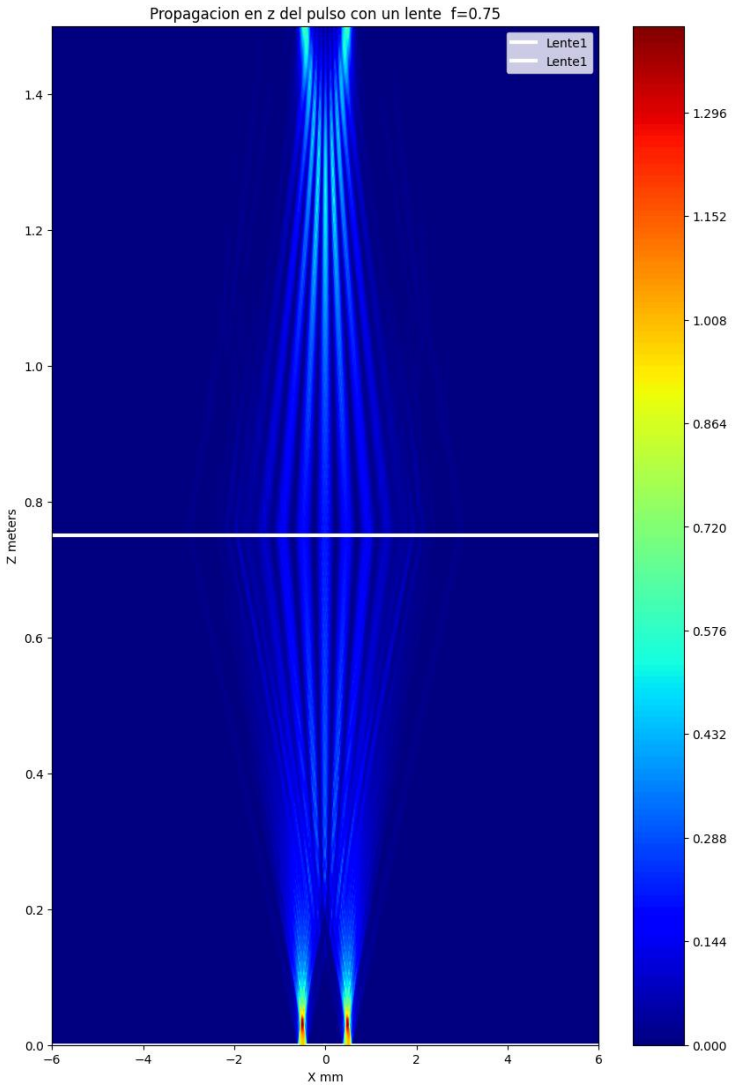


Figura 1616. Propagación a lo largo de z con una lente en $z = 0.75$ m

Luego, se grafica el error relativo la energía del frente de onda en función de la distancia de propagación z. Nuevamente, se encuentra que la propagación cuenta con un error mucho menor a 0.001.

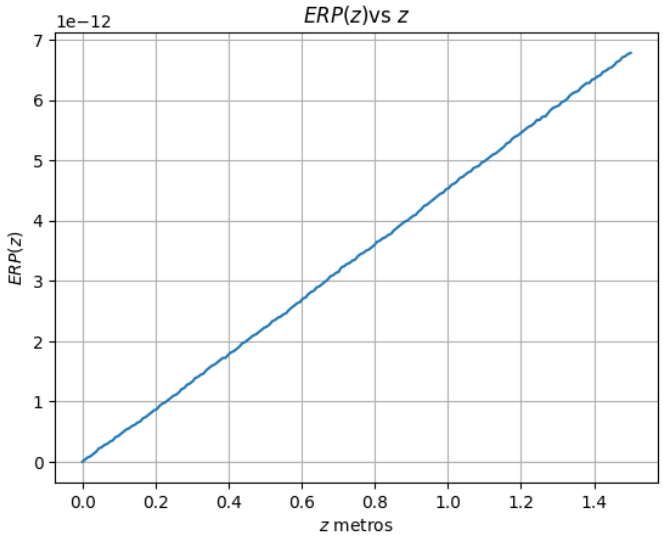


Figura 1717. *ERP(z) vs z propagación con lente*

Al mostrar las diferentes gráficas empalmadas, también se demuestra visualmente que la propagación ha sido correcta.

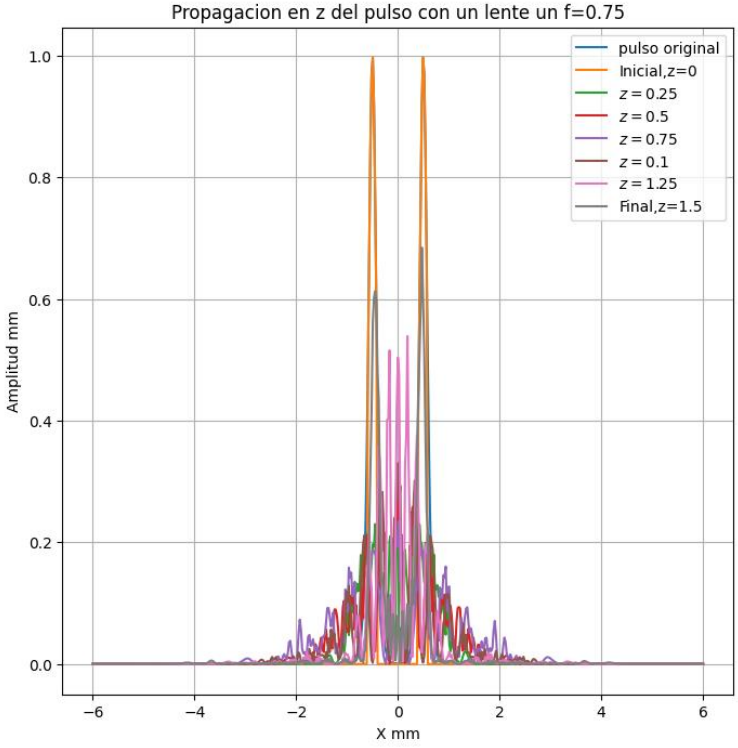


Figura 18. Propagación con lente en diferentes z **18**

VII. Simulación de pantalla con una transmitancia diferente

Para este último, caso se propone un campo inicial que oscile de la forma,

$$U_{i,1} = \sin(k\,x)\, e^{\frac{-x^2}{k_1}},$$

donde $k_1 \equiv 1 * 10^{-3} mm$. Se puede ver su propagación en la Figura 19.

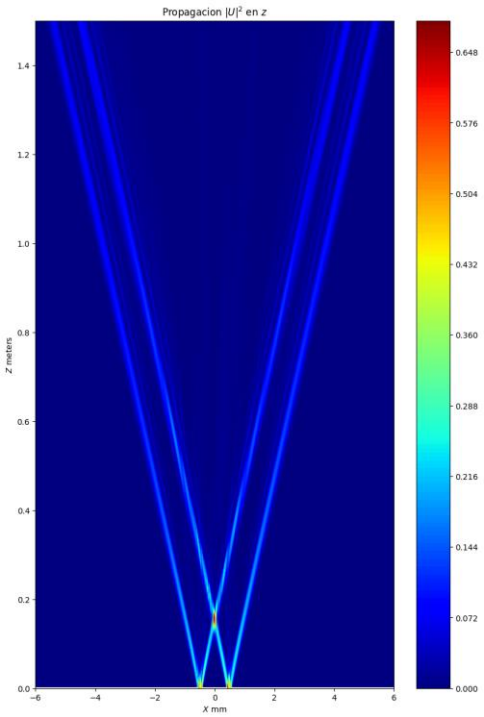


Figura 19. Propagación segundo caso de una onda *19*

Para simular la nueva lente, se utiliza una nueva transmitancia de la forma

$$T_l = \frac{f}{3}\cos^3(f\,k\,x)$$

con una distancia focal

$$f = 0.4\,m.$$

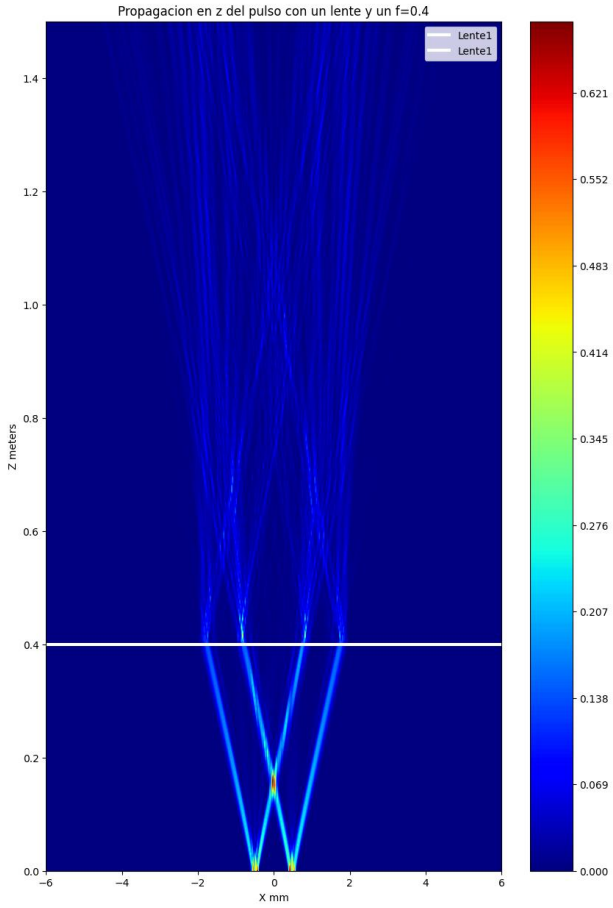


Figura 2020. Propagación segundo caso con un lente $f = 0.4$

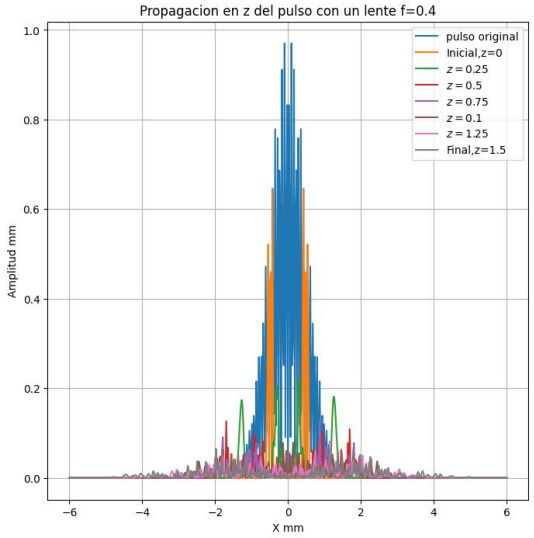


Figura 2121. Propagación en distintos puntos de z

VIII. Discusión final

Arif:

Al probar con diferentes condiciones iniciales observamos que entro los mayore contribuyentes a la propagación son las rendijas y sus dimensiones. Desde un principio en el que se probó el código en el cual no se tenían rendijas, los pulsos simplemente se atenuaban y no interferían entre sí. Después al incluir las rendijas observamos como estas controlan el ángulo en el que se interfiere y forma la interferencia de la propagación. Asimismo, observamos que debido a lo estrecho de b, la propagación no se ve muy afectada por funciones en forma de pulsos como las gaussianas, hiperbólicas, elípticas, y funciones constantes con anchos mayores a la rendija. En el último apartado, se ve que se obtiene un patrón diferente al usar una función que oscila con una frecuencia $\frac{1}{\lambda}$ de tal modo que alcanzan a entrar algunas oscilaciones en la rendija. Además, algo importante a lo largo de las simulaciones fue el cálculo de la energía y del error, que como se ve en la figura 10 y 11 la energía se conserva y el error es muy por debajo de la tolerancia.

Al aumentar el número de puntos obtenemos una mayor precisión en la propagación, sin embargo, en estos casos, se daban casos en los que los que los haces de luz se reflejan como si fuesen espejos los bordes; dando pie a errores numéricos, ya que no se debería de reflejar, en dichos casos lo que habría de hacer es atenuar esas reflexiones o hacer que las paredes absorban y no reflejen.

A lo largo del desarrollo de la actividad se presentaron algunas dificultades, uno de los principales es que en algunos casos los pulsos se reflejaban o que la propagación de la lente en un principio no enfocaba. Asimismo, otro de los mayores problemas a lo largo de la actividad fue el calcular las distancias entre las franjas que se generaban en la propagación. Esto debido a que se tenía que calcular los picos o puntos máximos de las oscilaciones. Este proceso fue de gran dificultad ya que la función oscilaba dentro de sí misma lo que dificulto encontrar su máximo. Sin embargo, este problema se solución a través de la suavización de la función y sus derivadas, así como del uso de la librería scipy. No obstante, el usar aproximaciones a través de la suavización, implico que se cargara con un nuevo error numérico. Esto se hace presente en la verificación del primer apartado en el inciso *d) del* $\Delta\theta$, *donde* el error se debe en parte a ser un error numérico, además debido a la suavización de las ondas.

En este proyecto aprendimos acerca del experimento de Young pudimos comparar y confirmar los resultados teóricos con nuestros resultados numéricos. En este proyecto exploramos un poco acerca el régimen paraxial y cómo es que se aplica a la propagación de ondas. Considero que gracias a este proyecto pude comprender de mejor manera el comportamiento de la luz y su propagación también considero haber desarrollado más a fondo mis habilidades de lógica y programacion que serán de gran ayuda en cursos siguientes.

Alberto

Al cambiar las condiciones iniciales del inciso 1, se encuentra que, dependiendo de la forma del pulso, se obtenían diferentes propagaciones, las cuales podían ser más o menos

estables. Por ejemplo, al propagar un pulso Gaussiano o suave, la forma de este se mantenía durante todo el trayecto. Al añadir la rendija, el pulso se dispersó; sin embargo, si la rendija no era lo suficientemente pequeña y esta no bloqueaba gran parte del pulso, entonces los pulsos se propagaban como en las Figuras 19 y 20, como si dos haces surgieran de la rendija.

La realización del proyecto trajo consigo ciertas dificultades en sus diferentes etapas. Por ejemplo, en el problema 1, si se cambiaba el rango o el número de puntos transversales, el pulso podía dispersarse a los laterales y reflejarse por errores numéricos presentes en el algoritmo, ingresando ruido visual a la simulación. También, para cálculos de comprobación, como en el caso de la separación angular, fue necesario realizar una suavización de los valores de la simulación para obtener una gráfica de la intensidad relativa que reflejase más aquella descrita por las ecuaciones 1 y 2 (Figura 2).

En este proyecto, aprendimos sobre la formación y el comportamiento de los patrones de interferencia creados cuando diferentes pulsos de ondas de luz interactúan entre sí. También pudimos explorar el fenómeno de difracción al incidir una onda con una rendija, a través de la ecuación paraxial de onda.

En cuanto a habilidades técnicas, hemos puesto en práctica nuestras habilidades en programación, específicamente en Python, para implementar las rutinas de simulación, especialmente utilizando librerías científicas y matemáticas como *Numpy* y *Scipy*. Además de sólo practicar nuestra lógica computacional y de programación, hemos adquirido habilidades relacionadas con la simulación de sistemas físicos, comprendiendo en buena proporción el cómo se modela y simula la propagación de ondas electromagnéticas bajo ciertas condiciones de cierto experimento. Por último, mediante diferentes métricas y comprobaciones analizamos y comparamos resultados teóricos con aquellos numéricos calculados para evaluar la precisión y validez de los algoritmos de simulación implementadas.

Referencias

[1] Gutiérrez-Vega, J. C. (2023, junio). *Notes on the Numerical Solution of the Diffusion Equation with the Finite Difference Method*.

Apéndice

In [112…
```
import sympy as sp
import numpy as np
from matplotlib import pyplot as plt
from matplotlib.animation import FuncAnimation
from IPython import display
from mpl_toolkits.mplot3d import Axes3D
from IPython.display import HTML
from scipy import special
```

In [194…
```
nx=511
nz=200
xf=0.006
xs=np.linspace(-xf,xf,nx)
xss=xs*1000
dx=np.diff(xs)[0]
zf=1.5
zs=np.linspace(0,zf,nz)
dz=np.diff(zs)[0]
X,Z=np.meshgrid(xss,zs)
```

$$\frac{\partial^2 U}{\partial x^2} + i2k\frac{\partial U}{\partial z} = 0$$

reacomodando

$$-\frac{1}{i2k}\frac{\partial^2 U}{\partial x^2} = \frac{\partial U}{\partial z}$$

$$\therefore \kappa = -\frac{1}{i2k}$$

$$\alpha = \kappa\frac{\Delta z}{\Delta x^2}$$

$$k = \frac{2\pi}{\lambda}$$

In [194…
```
l=632.8*1e-9
k=(2*np.pi/l)
kappa=-1/(2*1j*k)

a=kappa*dz/(dx**2)
h=0.001

b=0.0002#0.2 mm

##C-inversa

v1=np.ones(nx)*2*(1+a)
v2=np.ones(nx-1)*(-a)
C0=np.diag(v1,k=0)
C1=np.diag(v2,k=1)
```

```
C2=np.diag(v2,k=-1)

C=np.linalg.inv((C1+C2+C0))
#D
v3=np.ones(nx)*2*(1-a)
D1=np.diag(v3,k=0)

D=(-C1-C2+D1)

P=np.matmul(C,D)

nxx=int(nx/2)
U=np.zeros((nz,nx))*1j
Ul=np.zeros(nx)
```

Pulso inicial y condiciones de frontera

In [194…

```
[sn, cn, dn, phi] = special.ellipj(xs*1000, 0.7);
#plt.plot(xss,cn)

#po=np.real(np.exp(1j*((k)*xs)))#+np.real(np.exp(1j*((k+b)*xs)))#=np.cos(1*xs)#2/np
#po=np.exp((-(xs)**2)/0.01)
#po=np.exp((-(xs-h/2)**2)/0.0000001)+np.exp((-(xs+h/2)**2)/0.0000001)
#po=np.exp((-(xs)**2)/0.0000001)
#po=dn
po=1/np.cosh((xs-h/2)/0.0001)+1/np.cosh((xs+h/2)/0.0001)
#po=np.cos(k*xs)
#po=np.exp(-((xs)**2)/0.000001)*np.sin(k*xs)


U[:,0]=0
U[:,-1]=0

U[0,:]=po
#Barrera izquiera
U[0,0:nxx-int((h/(2*dx)).round(0))-int((b/(2*dx)).round(0))]=0

#barrera derecha
U[0,nxx+int((h/(2*dx)).round(0))+int((b/(2*dx)).round(0)):-1]=0
U[0,-1]=0

# Barrera Central
U[0,nxx-int(h/(2*dx))+int(b/(2*dx)):nxx]=0
U[0,nxx:nxx+int((h/(2*dx)).round(0))-int((b/(2*dx)).round(0))]=0
```

Propagacion en z del pulso

In [194…

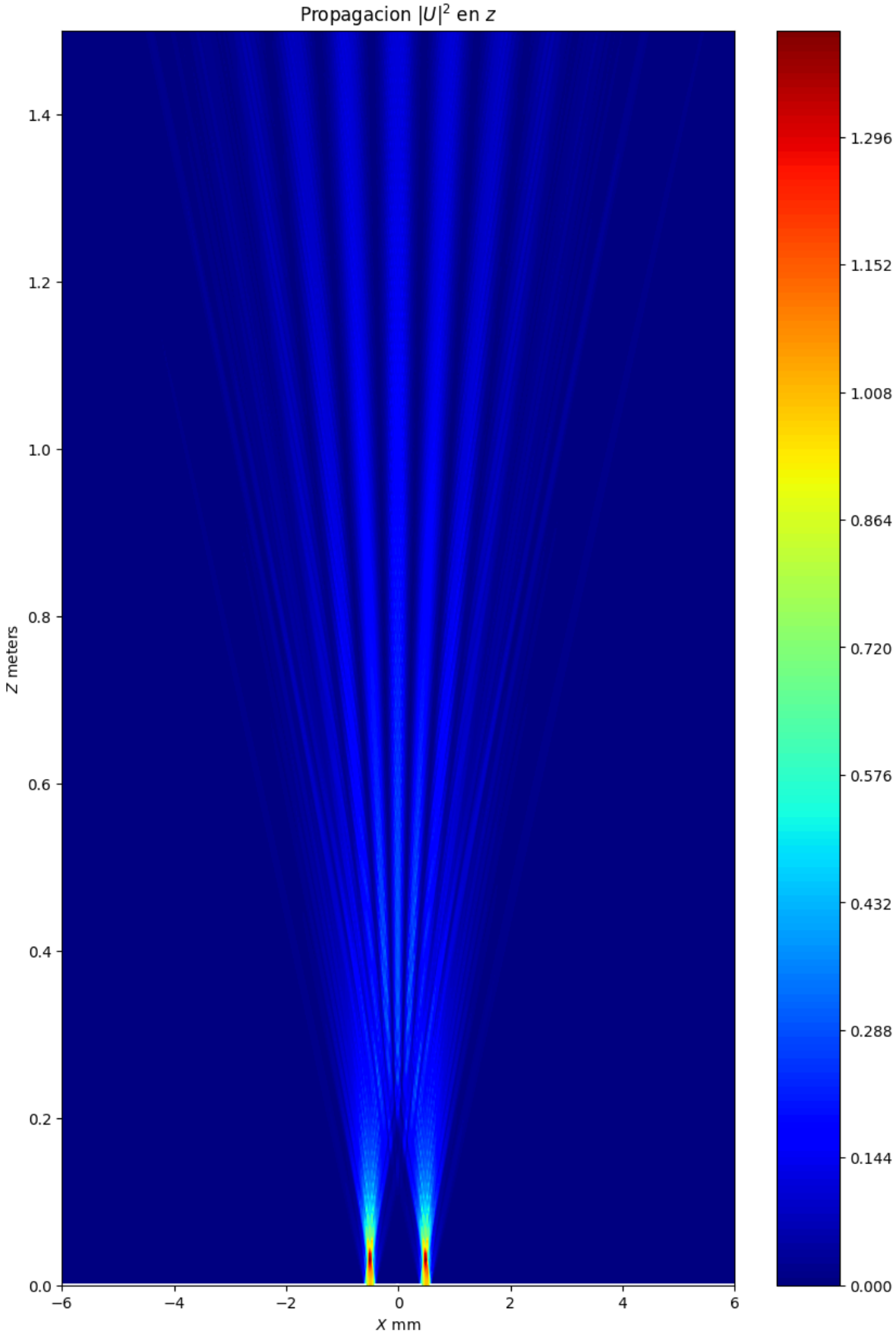
```
for i in range(0,nz-1):
    us=np.matmul(P,U[i,:])
    U[i+1,:]=us
```

In [194…

```
fig=plt.figure(figsize=(10,15))
```

```
ax = fig.add_subplot(111)
ax.clear()
prop=ax.contourf(X,Z,np.abs(U)**2,levels=200,cmap='jet')
plt.xlim([-6,6])
plt.colorbar(prop)
lwd=2.6
col='white'
#plt.xlim([-xf-2,xf+2])
ax.plot(xss[0:nxx-int(h/(2*dx))-int(b/(2*dx))],xss[0:nxx-int(h/(2*dx))-int(b/(2*dx)
ax.plot(xss[nxx+int(h/(2*dx))+int(b/(2*dx)):-1],xss[nxx+int(h/(2*dx))+int(b/(2*dx))
ax.plot(xss[nxx-int(h/(2*dx))+int(b/(2*dx)):nxx],xss[nxx-int(h/(2*dx))+int(b/(2*dx)
ax.plot(xss[nxx:nxx+int(h/(2*dx))-int(b/(2*dx))],xss[nxx:nxx+int(h/(2*dx))-int(b/(2
#plt.title('Propagacion abs(U)^2 en z')
plt.title(r'Propagacion $|U|^{2}$ en $z$', fontsize='large')
plt.xlabel(r'$X$ mm')
plt.ylabel(r'$Z$ meters')
```

Out[1947]: Text(0, 0.5, 'Z meters')

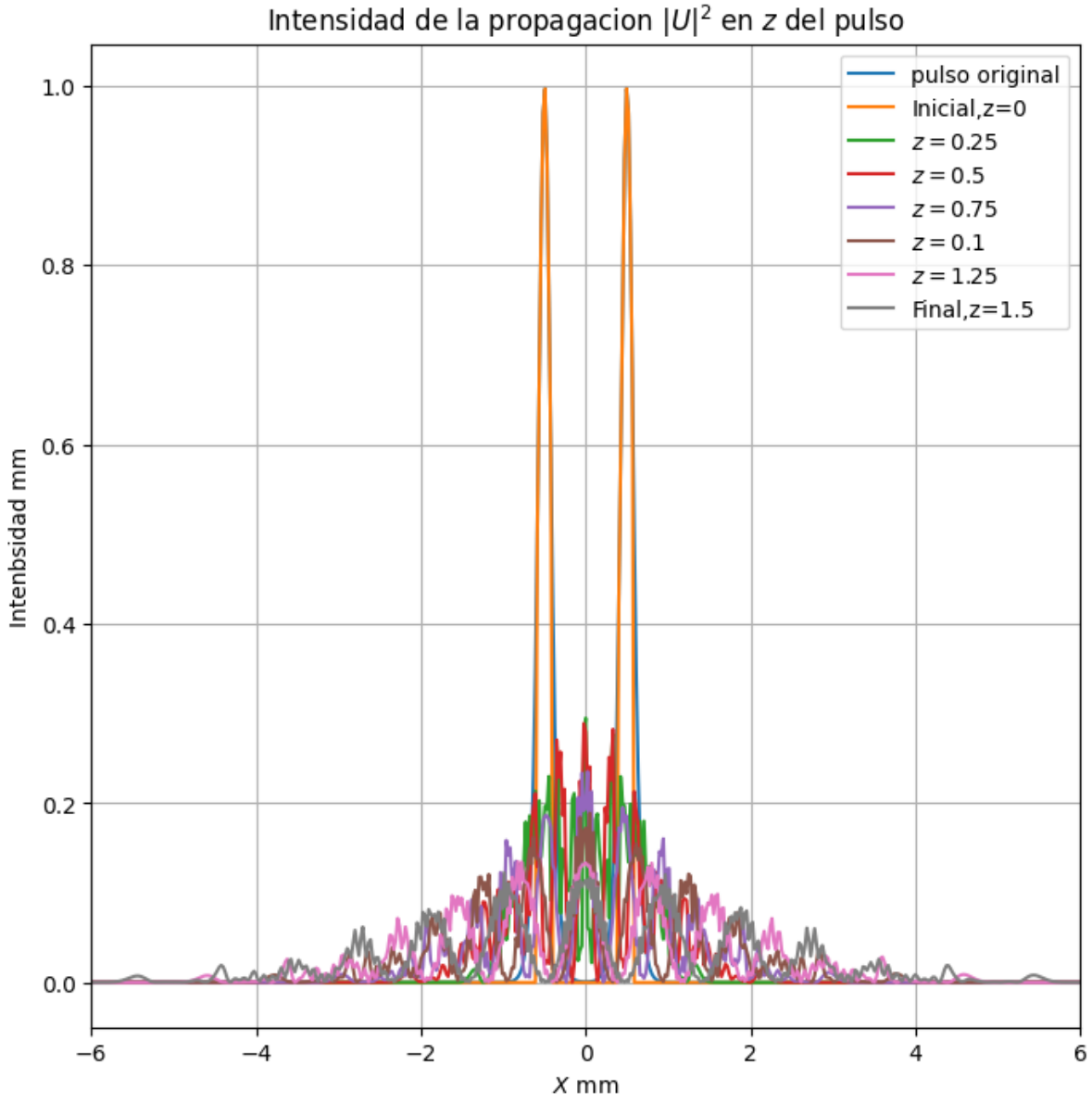


Empalme Graficas unidimensionales

In [194…

```
fig2=plt.figure(figsize=(8,8))
plt.plot(xss,po**2,label='pulso original')
plt.plot(xss,np.abs(U[0,:])**2,label='Inicial,z=0')
plt.plot(xss,np.abs(U[int(nz/6)*1,:])**2,label='$z=0.25$')
plt.plot(xss,np.abs(U[int(nz/6)*2,:])**2,label='$z=0.5$')
plt.plot(xss,np.abs(U[int(nz/6)*3,:])**2,label='$z=0.75$')
plt.plot(xss,np.abs(U[int(nz/6)*4,:])**2,label='$z=0.1$')
plt.plot(xss,np.abs(U[int(nz/6)*5,:])**2,label='$z=1.25$')
plt.plot(xss,np.abs(U[-1,:])**2,label='Final,z=1.5')

#plt.xlim([min(xss),max(xss)])
plt.xlim([-xf*1000,xf*1000])
#plt.axis('equal')
plt.grid('on')
plt.title(r'Intensidad de la propagacion $|U|^{2}$ en $z$ del pulso')
plt.xlabel('$X$ mm')
plt.ylabel('Intenbsidad mm')
plt.legend()
```

Out[1948]: <matplotlib.legend.Legend at 0x1acb6ab2d00>

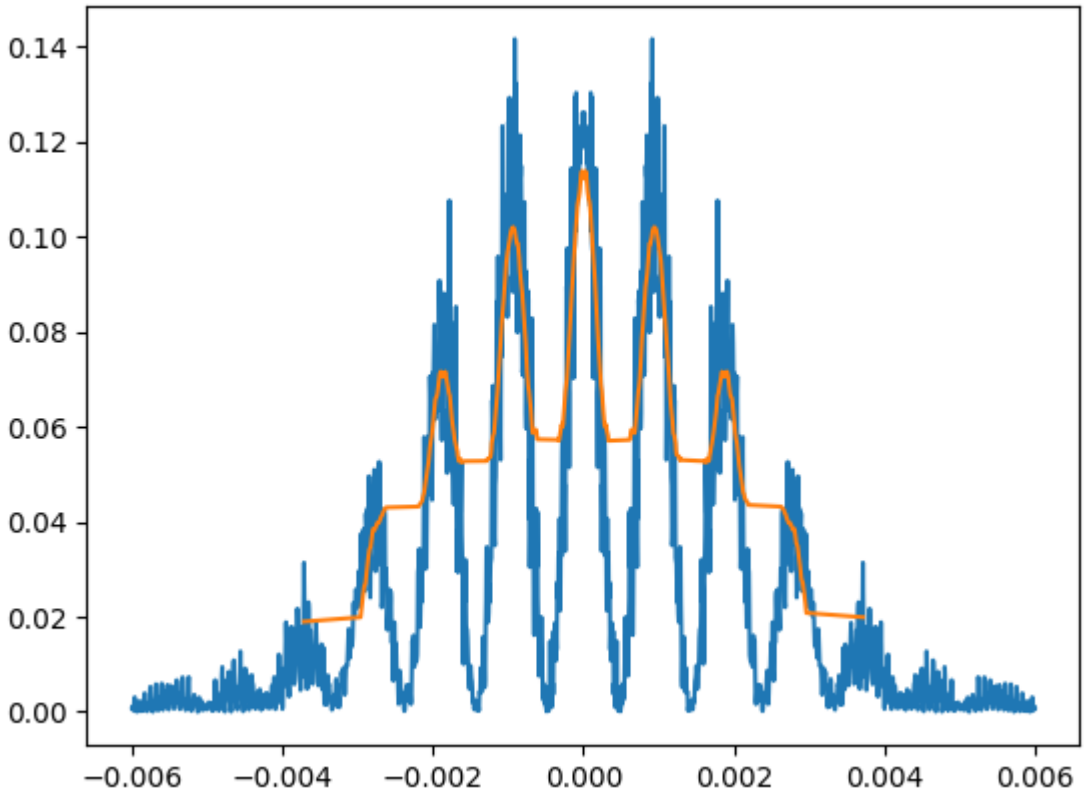


Derivada

In [185…
```
Us=abs(U[-1,:])**2
iss=np.where(abs(U[-1,:])**2>0.03)
```

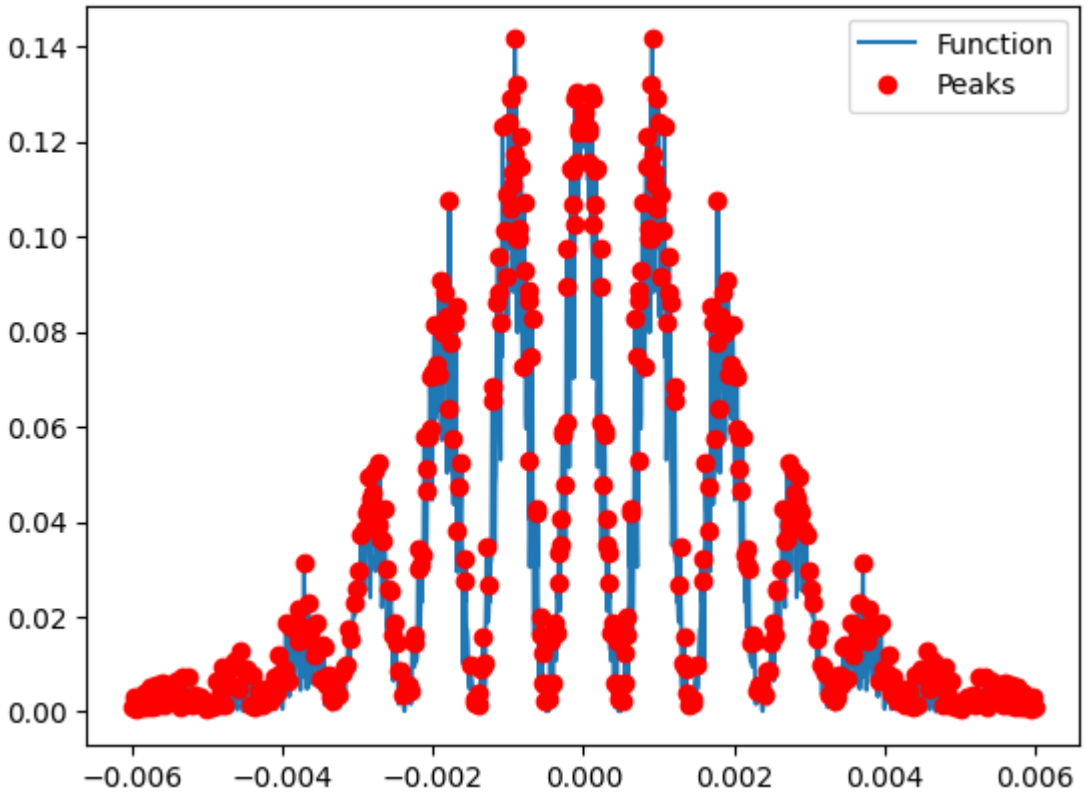
In [186…
```
window_size = 50#50
y_smooth = np.convolve(Us[iss], np.ones(window_size)/window_size, mode='same')
plt.plot(xs,abs(U[-1,:])**2)
plt.plot(xs[iss],y_smooth)
```

Out[1860]: [<matplotlib.lines.Line2D at 0x1ac8bc8dd00>]



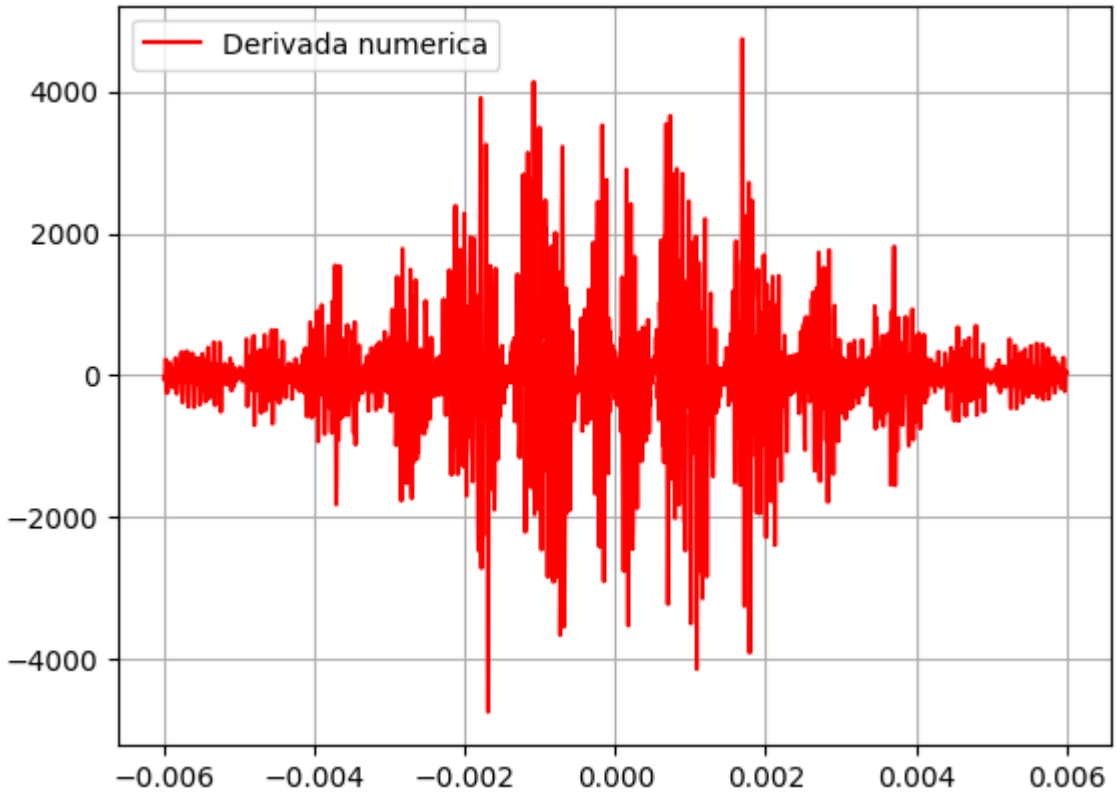
In [170…

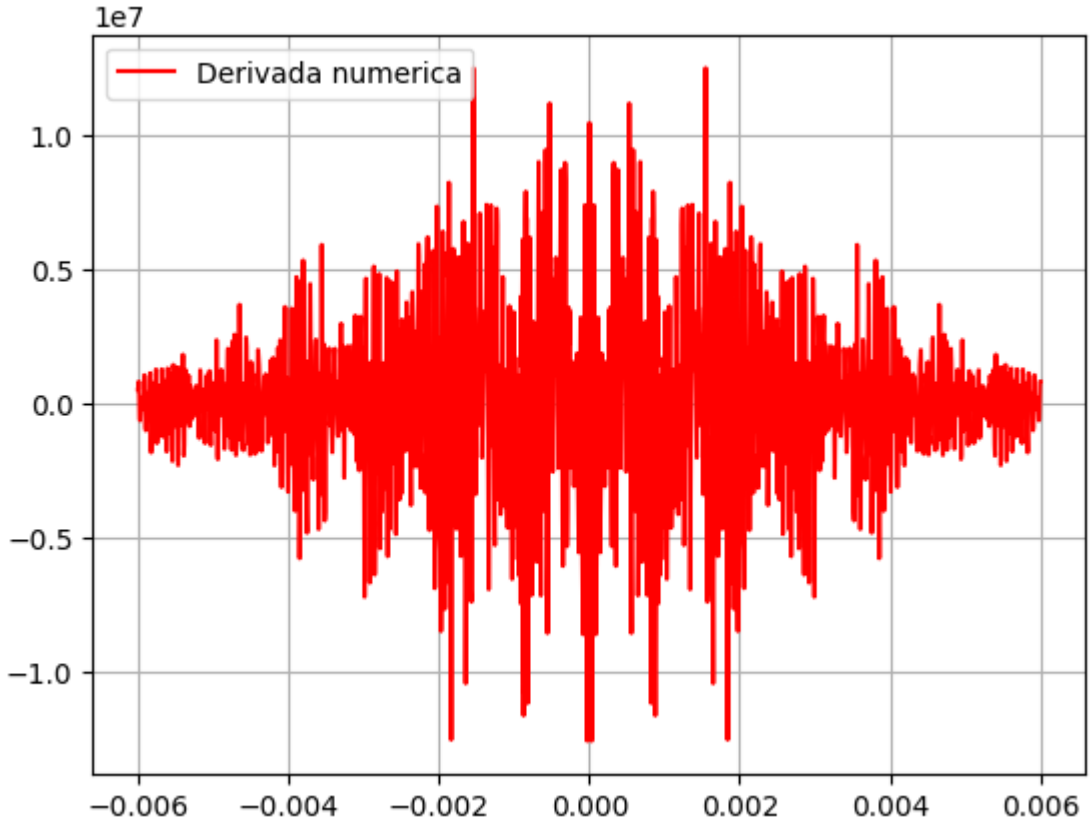
```
from scipy.signal import argrelextrema
#import matplotlib.pyplot as plt

# Example discrete oscillating function (replace with your own data)
x = xs
y = Us

# Find local maxima and minima indices
lm = argrelextrema(y, np.greater)[0]
local_minima_indices = argrelextrema(y, np.less)[0]

# Plot original function with peaks
plt.plot(x, y, label='Function')
plt.plot(x[lm], y[lm], 'ro', label='Peaks')
#plt.plot(x[local_minima_indices], y[local_minima_indices], 'bo', label='Troughs')
plt.legend()
plt.show()

lmm=np.where(Us[lm]>0.04)[0]
lm=lm[lmm]
#plt.scatter(xs[lm],xs[lm]/xs[lm])
```

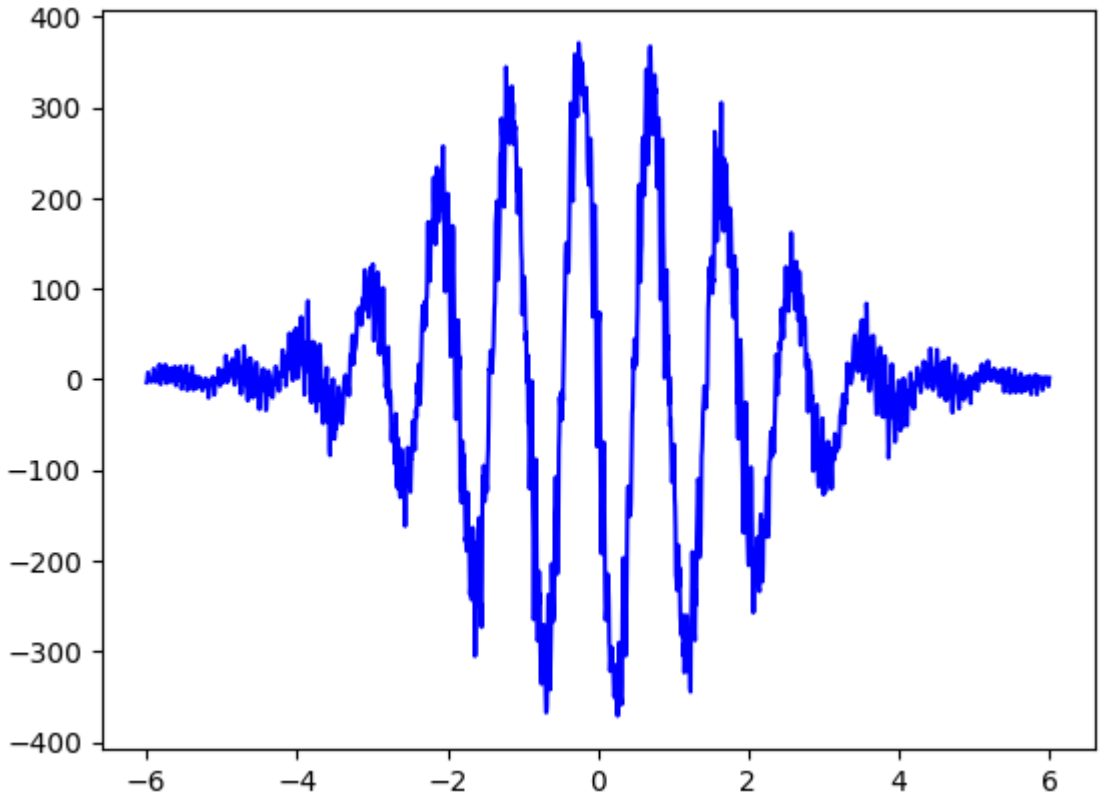


In [173…

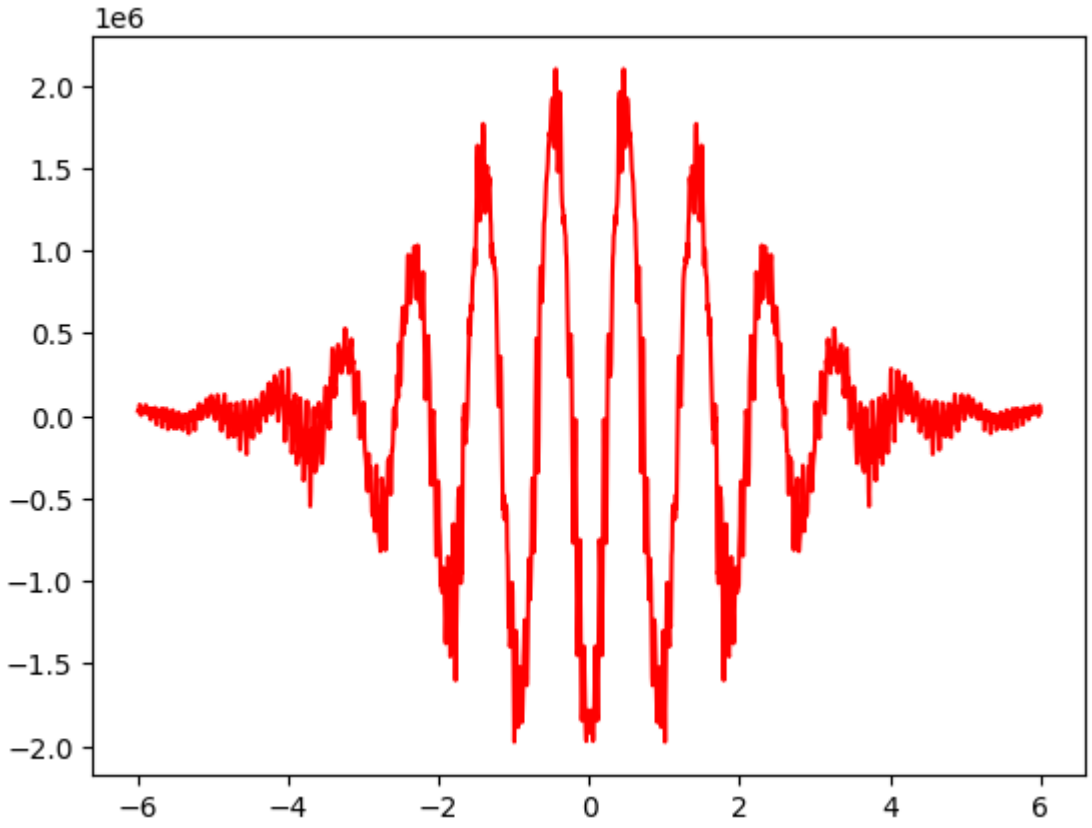
```
Us1=Us[iss]
pd=df(Us,dx,-xf,xf)## Derivada

window_size = 50#85
pd = np.convolve(pd, np.ones(window_size)/window_size, mode='same')

sd=df(pd,dx,-xf,xf)## Derivada
sd = np.convolve(sd, np.ones(window_size)/window_size, mode='same')
jim=np.where(sd < 0)[0]#Lugares en los que son negativas la segunda derivada
s=np.sign(pd)
xx=np.where(s[:-1]!=s[1:])[0]

xx1=xs[np.intersect1d(np.intersect1d(jim, xx),lm)]
#xx1=np.delete(xx1,5)
#xx1[3]=0
#xx1=np.delete(xx1,4)
dxx=np.diff(xx1)
#th=np.arctan(dxx/zf)
plt.plot(xss,pd,color='blue')
plt.show()
plt.plot(xss,sd,color='red')
#plt.grid('on')
```


Derivada numerica
4000
2000
0
−2000
−4000
−0.006
−0.004
−0.002
0.000
0.002
0.004
0.006


1e7
Derivada numerica
1.0
0.5
0.0
−0.5
−1.0
−0.006
−0.004
−0.002
0.000
0.002
0.004
0.006



Out[1736]: [<matplotlib.lines.Line2D at 0x1ac65163460>]



In [173…

```
xx1
xx1=np.delete(xx1,1)
xx1=np.delete(xx1,-1)
```

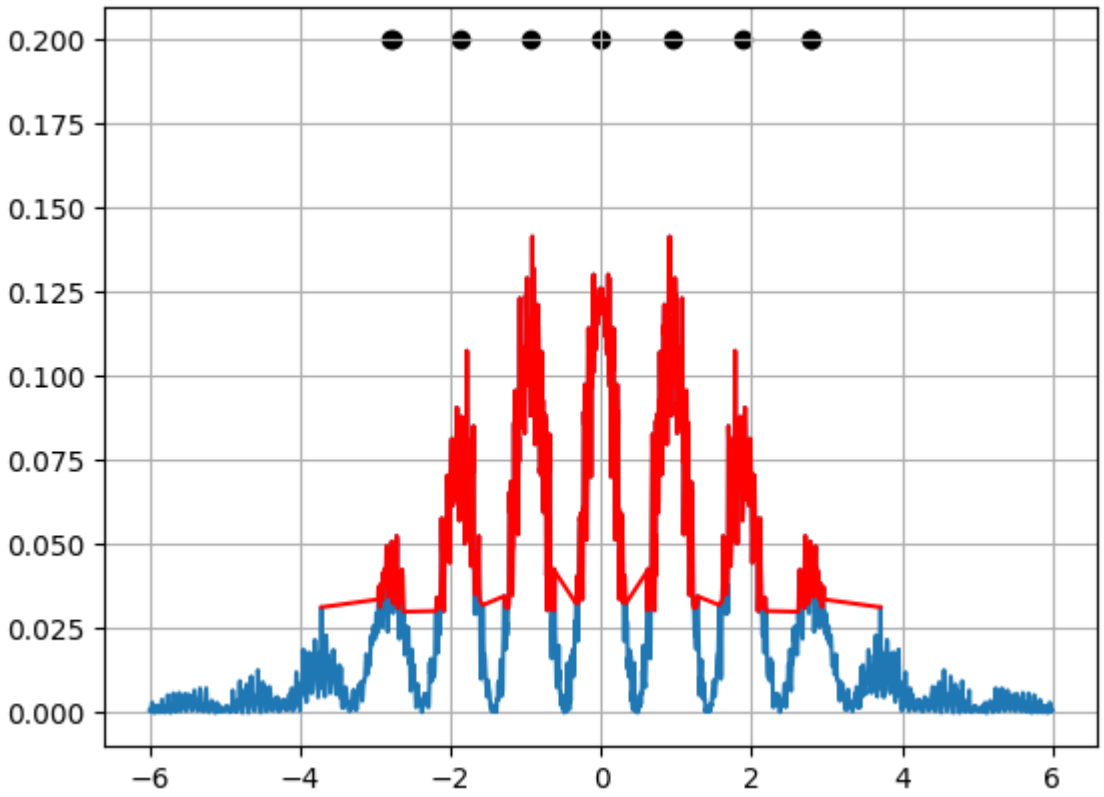
In [173…
```
np.mean(np.arctan(np.diff((xx1)/zf)))
```

Out[1738]: 0.0006200031836974097

In [173…
```
(np.mean(np.arctan(np.diff(xx1)/zf))-(l/h))/(l/h)*100
```

Out[1732]: -26.284636993423405

In [170…
```
plt.plot(xss,abs(U[-1,:])**2)
plt.plot(xss[iss],Us[iss],color='red')

plt.scatter(xx1*1000,np.ones(len(xx1))*0.2,color='black')
plt.grid('on')
```



In [138…
```
def df(y,dt,ti,tf):
    t=np.arange(ti,tf,dt)
    n=len(y)
    v1=np.ones(n-1)*(1/2)
    v2=np.ones(n-1)*(-1/2)
    D1=np.diag(v1,k=1)
    D2=np.diag(v2,k=-1)
    D=(D1+D2)/dt
    yp=np.matmul(D, y)
    yp[0]=yp[1]
    yp[-1]=yp[-2]
    ##Plot#################################################################
    #plt.plot(t,y,color='black',label='funcion')
    plt.plot(t,yp,color='red',label='Derivada numerica')
    #plt.axis('equal')
    plt.grid('on')
```

```
    plt.legend()
    plt.show()
    return yp
```

Teorico

$$\Delta\theta = \frac{\lambda}{h}$$

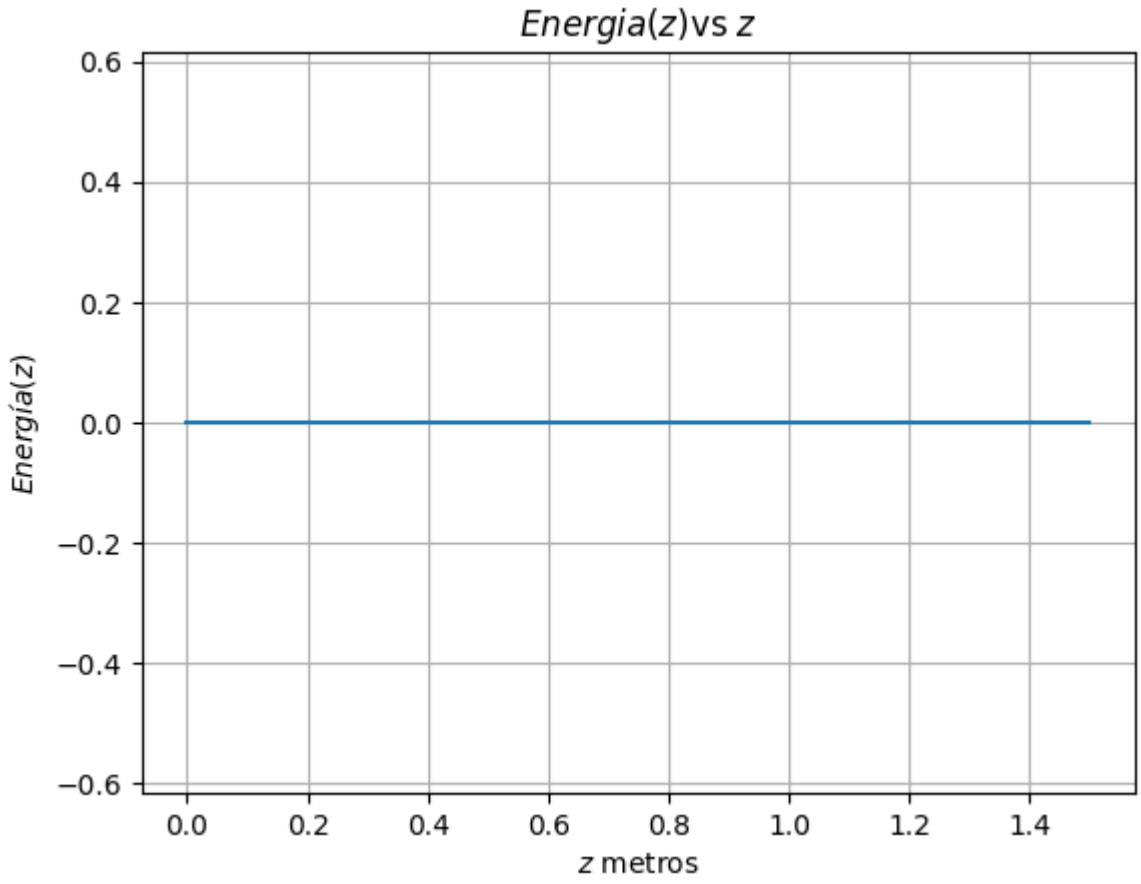
In [168…
```
l/h
```

Out[1682]: 0.0006328

$$Energía(z) \propto \int U^2 \approx \sum_j |U_j|^2 dx$$

In [194…
```
E=(abs(U)**2).sum(axis=1)*dx
plt.plot(zs,E)
plt.grid('on')
plt.title('$Energia(z)$vs $z$')
plt.xlabel('$z$ metros')
plt.ylabel('$Energía(z)$')
plt.axis('equal')
```

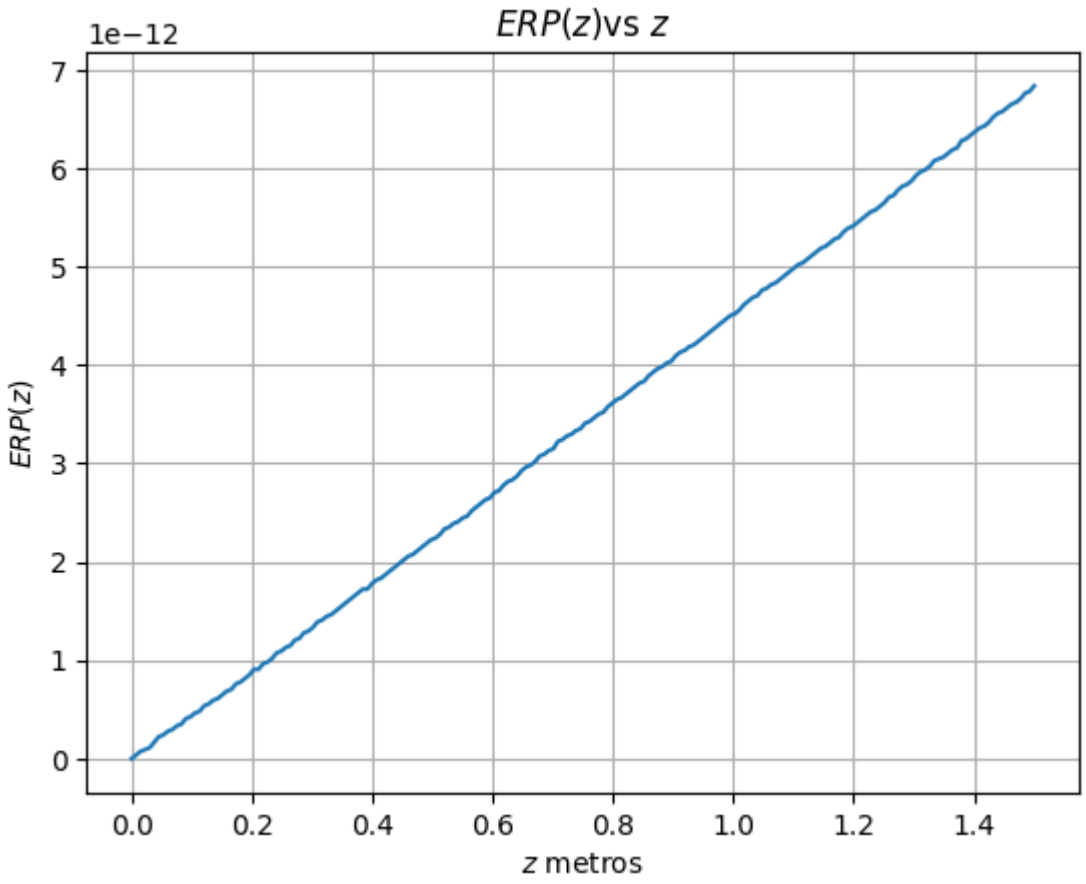
Out[1940]: (-0.07500000000000001, 1.575, 0.0002926655441124213, 0.00029266554411302845)



$$ERP(z) = 100 * |\frac{E_{inicial} - Energía(z)}{E_{inicial}}|$$

In [168…
```
plt.plot(zs,abs((E[0]-E)*100/E[0]))
plt.grid('on')
plt.title('$ERP(z)$vs $z$')
plt.xlabel('$z$ metros')
plt.ylabel('$ERP(z)$')
#plt.axis('equal')
```

Out[1684]: Text(0, 0.5, '$ERP(z)$')



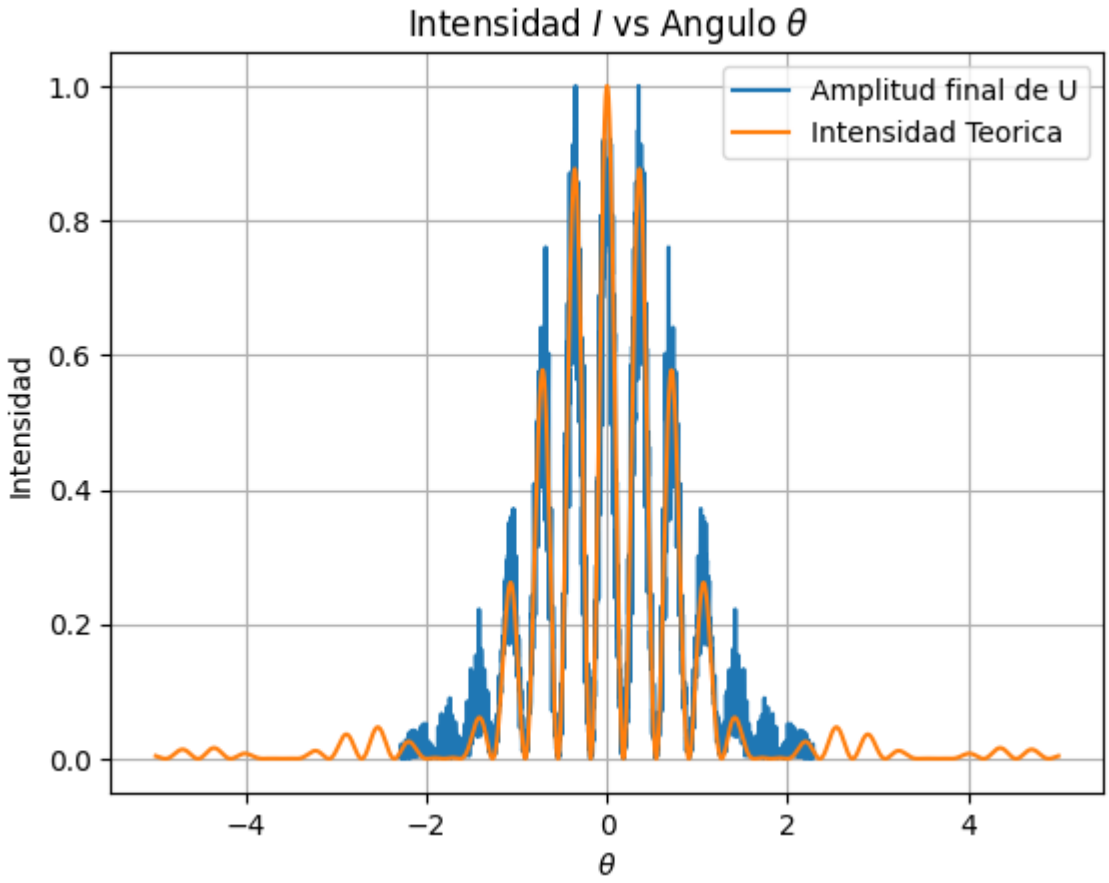
Intensidad

In [187…

```
#dth=l/h
deg=0.5
th1=np.linspace(-deg,deg,10000)*(np.pi/180) #####radianes
th=np.linspace(-deg*10,deg*10,10000)


B=(1/2)*k*b*np.sin(th1)
y=(1/2)*k*h*np.sin(th1)
I=((np.sin(B)/B)**2)*np.cos(y)**2

plt.plot(np.arctan(xs/zf)*(180/np.pi)*10,Us/max(Us),label='Amplitud final de U')
plt.plot(th,I,label='Intensidad Teorica')
window_size = 20

#plt.xlim([min(th1)/100,max(th1)/100])
#plt.ylim([0,1])
#Suavisamoiento

#plt.plot(xs,abs(U[-1,:])**2)
#plt.plot(np.arctan(xs/zf)*(180/np.pi)*10,Us/max(Us),label='Amplitud final de U')
#plt.plot(np.arctan(xss/zf),Us,label='normal')
#plt.axis('equal')
plt.title('Intensidad $I$ vs Angulo $\u03B8$')
plt.xlabel('$\u03B8$')
```

```
plt.ylabel('Intensidad')
plt.grid('on')
plt.legend()
```

Out[1875]: <matplotlib.legend.Legend at 0x1ab91ce5490>

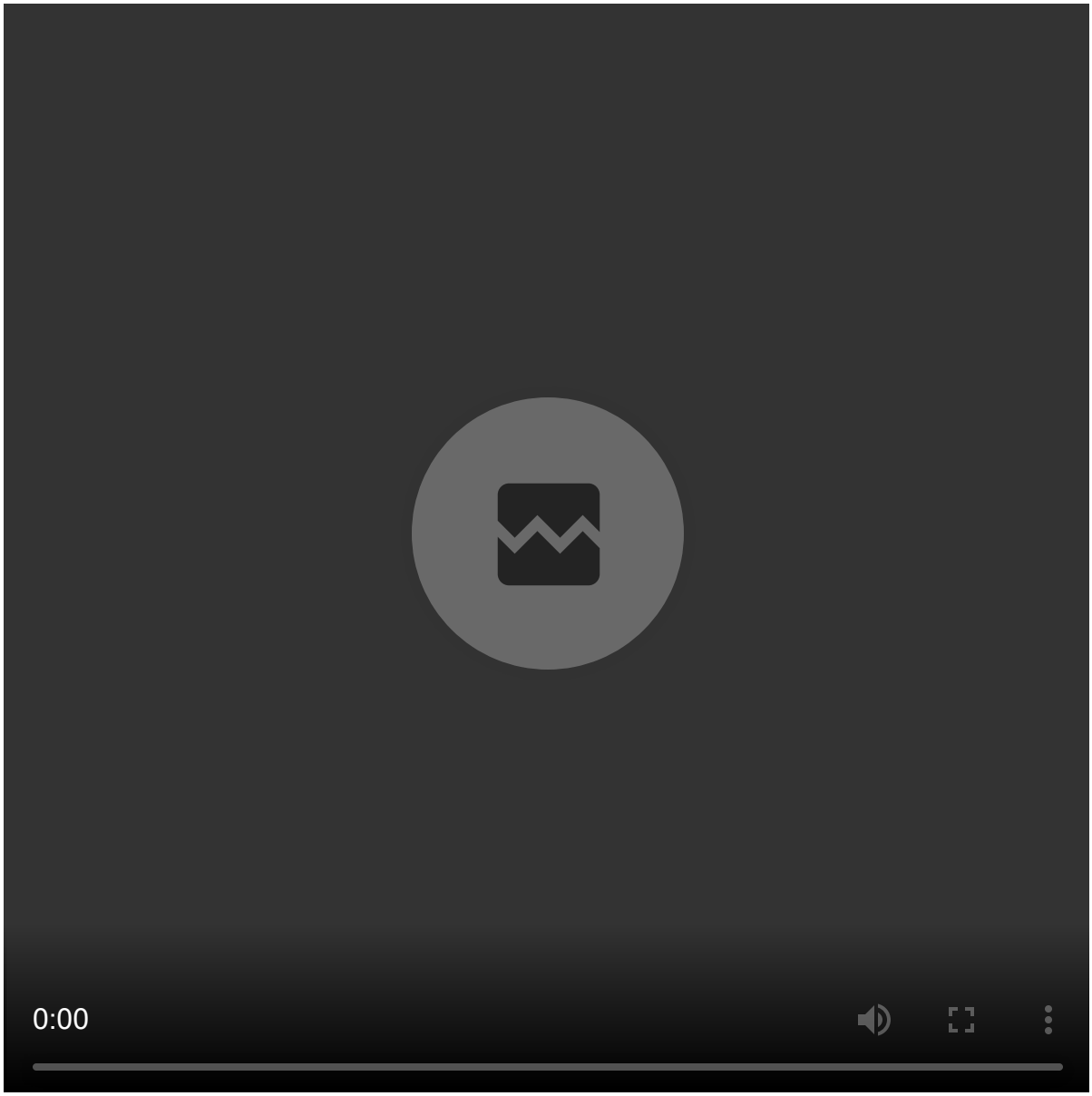


Animacion

In [124…

```
fig=plt.figure(figsize=(6,6))
ax = fig.add_subplot(111)
#plt.axis('equal')
plt.grid('on')

plt.xlim([-xf,xf])
plt.ylim([-1,1])

def animate(i):
    ax.clear()
    plt.xlim([-xf,xf])
    plt.ylim([-1,1])
    #plt.axis('equal')
    plt.grid('on')
    ax.plot(xs,np.abs(U[i,:])**2,color='red')
    #plt.legend()
    return fig

fr=np.linspace(0,nz,100,dtype=int)
```

```
anim=FuncAnimation(fig,animate,frames=nz,interval=24)
video=anim.to_html5_video()
html=display.HTML(video)
display.display(html)
plt.close()
```



2. Propagacion en z del pulso con una lente

Veamos el efecto de colocar una lente convergente a mitad de la distancia de propagación, ver la Fig. 1(b). Para esto propaga el campo como se hizo en el Prob. 1 pero ahora hasta z = 0.75 m. En ese lugar multiplica el campo por la transmitancia de una lente convergente

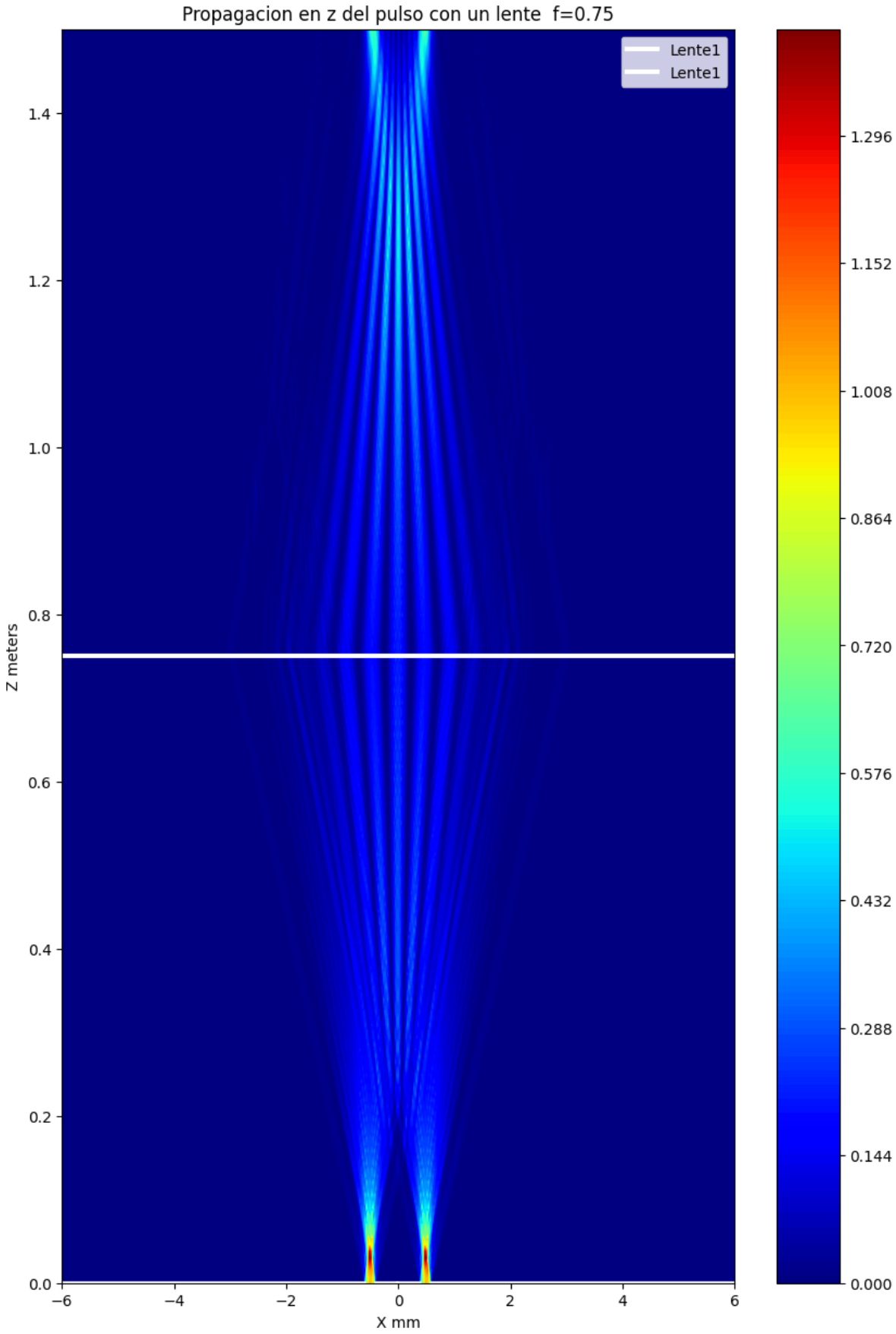
$$T_L(x) = exp(-i\frac{k}{2f}x^2),$$

$$f = 0.75m$$

In [193…
```
f=0.75
Tl=np.exp((-1j*k*xs**2)/(2*f))
for i in range(0,nz-1):
    if zs[i].round(2)==f:
         us=np.matmul(P,U[i,:])*Tl
    else:
        us=np.matmul(P,U[i,:])
    U[i+1,:]=us
```

In [193…
```
fig3=plt.figure(figsize=(10,15))
ax = fig3.add_subplot(111)
prop=ax.contourf(X,Z,np.abs(U)**2,levels=200,cmap='jet')
plt.colorbar(prop)

ff=np.ones(nx)*zff
ax.plot(xss,ff,1,color='white',lw=3,label='Lente1')
plt.legend()

lwd=2.6
col='white'
#plt.xlim([-xf-2,xf+2])
ax.plot(xss[0:nxx-int(h/(2*dx))-int(b/(2*dx))],xss[0:nxx-int(h/(2*dx))-int(b/(2*dx)
ax.plot(xss[nxx+int(h/(2*dx))+int(b/(2*dx)):-1],xss[nxx+int(h/(2*dx))+int(b/(2*dx))
ax.plot(xss[nxx-int(h/(2*dx))+int(b/(2*dx)):nxx],xss[nxx-int(h/(2*dx))+int(b/(2*dx)
ax.plot(xss[nxx:nxx+int(h/(2*dx))-int(b/(2*dx))],xss[nxx:nxx+int(h/(2*dx))-int(b/(2

plt.title('Propagacion en z del pulso con un lente  f='+str(f))
plt.xlabel('X mm')
plt.ylabel('Z meters')
```

Out[1937]: Text(0, 0.5, 'Z meters')



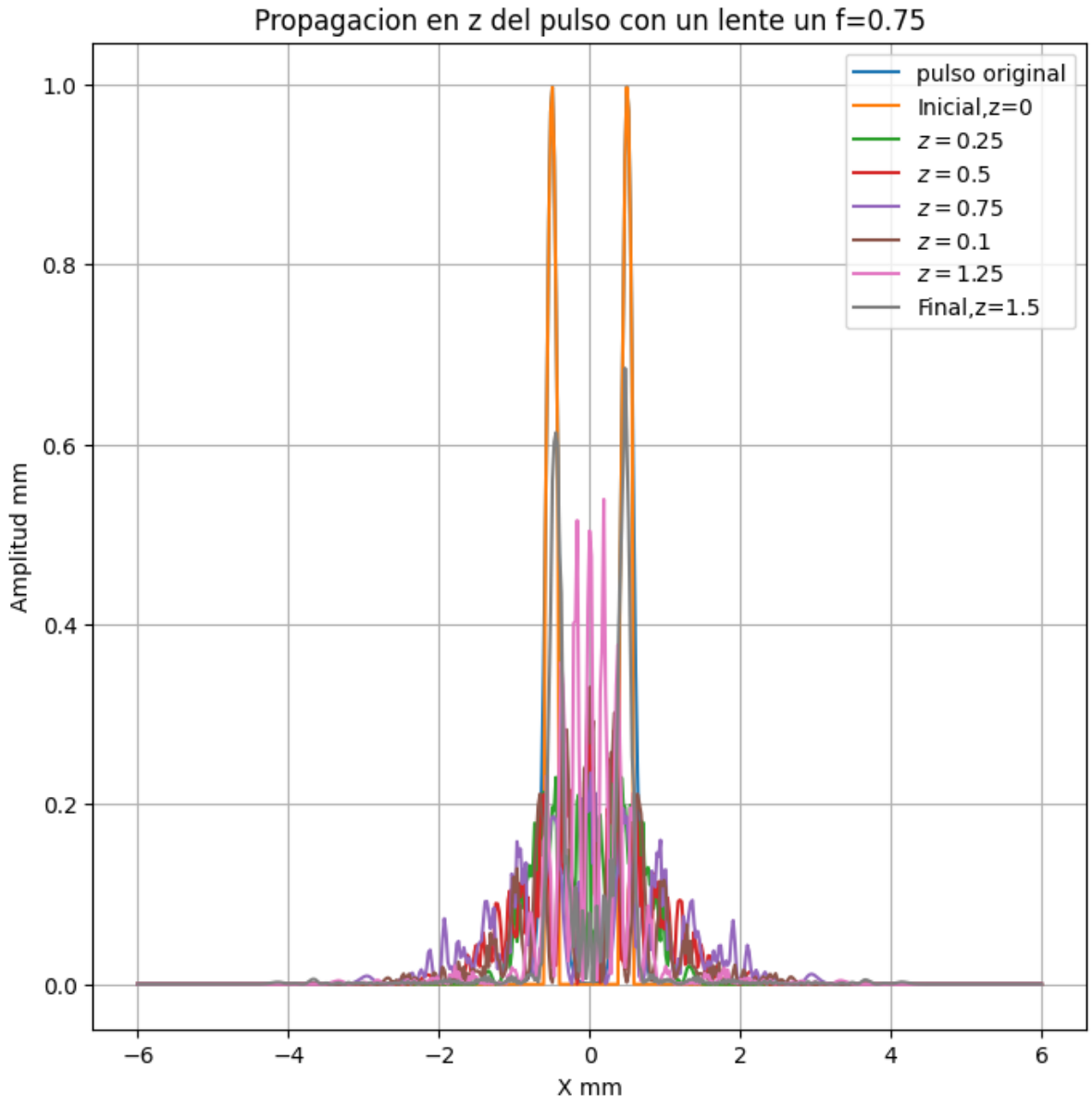
Empalme Graficas unidimensionales con

transmitancia lente

In [163…

```
fig4=plt.figure(figsize=(8,8))
plt.plot(xss,po**2,label='pulso original')
plt.plot(xss,np.abs(U[0,:])**2,label='Inicial,z=0')
plt.plot(xss,np.abs(U[int(nz/6)*1,:])**2,label='$z=0.25$')
plt.plot(xss,np.abs(U[int(nz/6)*2,:])**2,label='$z=0.5$')
plt.plot(xss,np.abs(U[int(nz/6)*3,:])**2,label='$z=0.75$')
plt.plot(xss,np.abs(U[int(nz/6)*4,:])**2,label='$z=0.1$')
plt.plot(xss,np.abs(U[int(nz/6)*5,:])**2,label='$z=1.25$')
plt.plot(xss,np.abs(U[-1,:])**2,label='Final,z=1.5')

plt.grid('on')
plt.title('Propagacion en z del pulso con un lente un f='+str(f))
plt.xlabel('X mm')
plt.ylabel('Amplitud mm')
plt.legend()
```

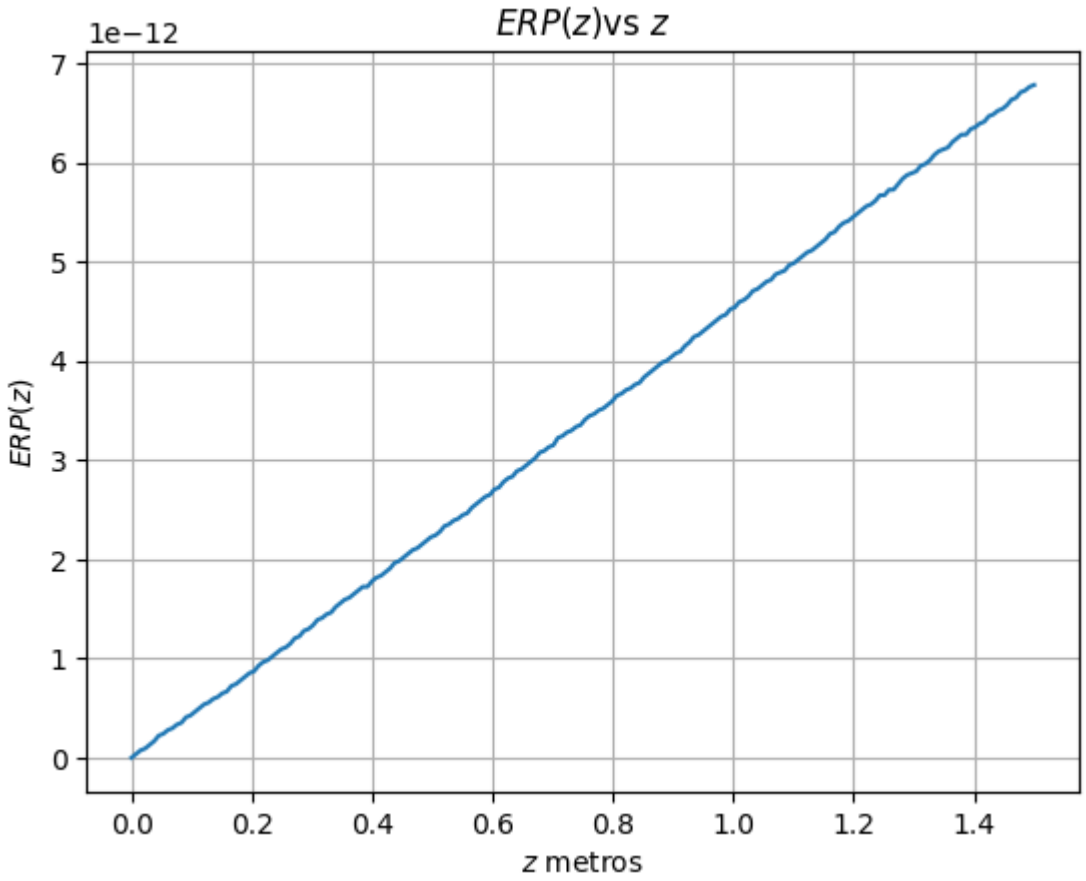
Out[1638]: <matplotlib.legend.Legend at 0x1ab77f5e4f0>



ERROR

In [163…
```
E=abs(U**2).sum(axis=1)*dx
#plt.plot(zs,E)
plt.plot(zs,abs((E[0]-E)*100/E[0]))
plt.grid('on')
plt.title('$ERP(z)$vs $z$')
plt.xlabel('$z$ metros')
plt.ylabel('$ERP(z)$')
#plt.axis('equal')
```

Out[1639]: Text(0, 0.5, '$ERP(z)$')



3. pantalla con una transmitancia diferente

In [188…
```
l=632.8*1e-9
k=(2*np.pi/l)
kappa=-1/(2*1j*k)

a=kappa*dz/(dx**2)
h=0.001

b=0.0002#0.2 mm

##C-inversa
```

```
v1=np.ones(nx)*2*(1+a)
v2=np.ones(nx-1)*(-a)
C0=np.diag(v1,k=0)
C1=np.diag(v2,k=1)
C2=np.diag(v2,k=-1)

C=np.linalg.inv((C1+C2+C0))
#D
v3=np.ones(nx)*2*(1-a)
D1=np.diag(v3,k=0)

D=(-C1-C2+D1)

P=np.matmul(C,D)

nxx=int(nx/2)
U=np.zeros((nz,nx))*1j
Ul=np.zeros(nx)
```

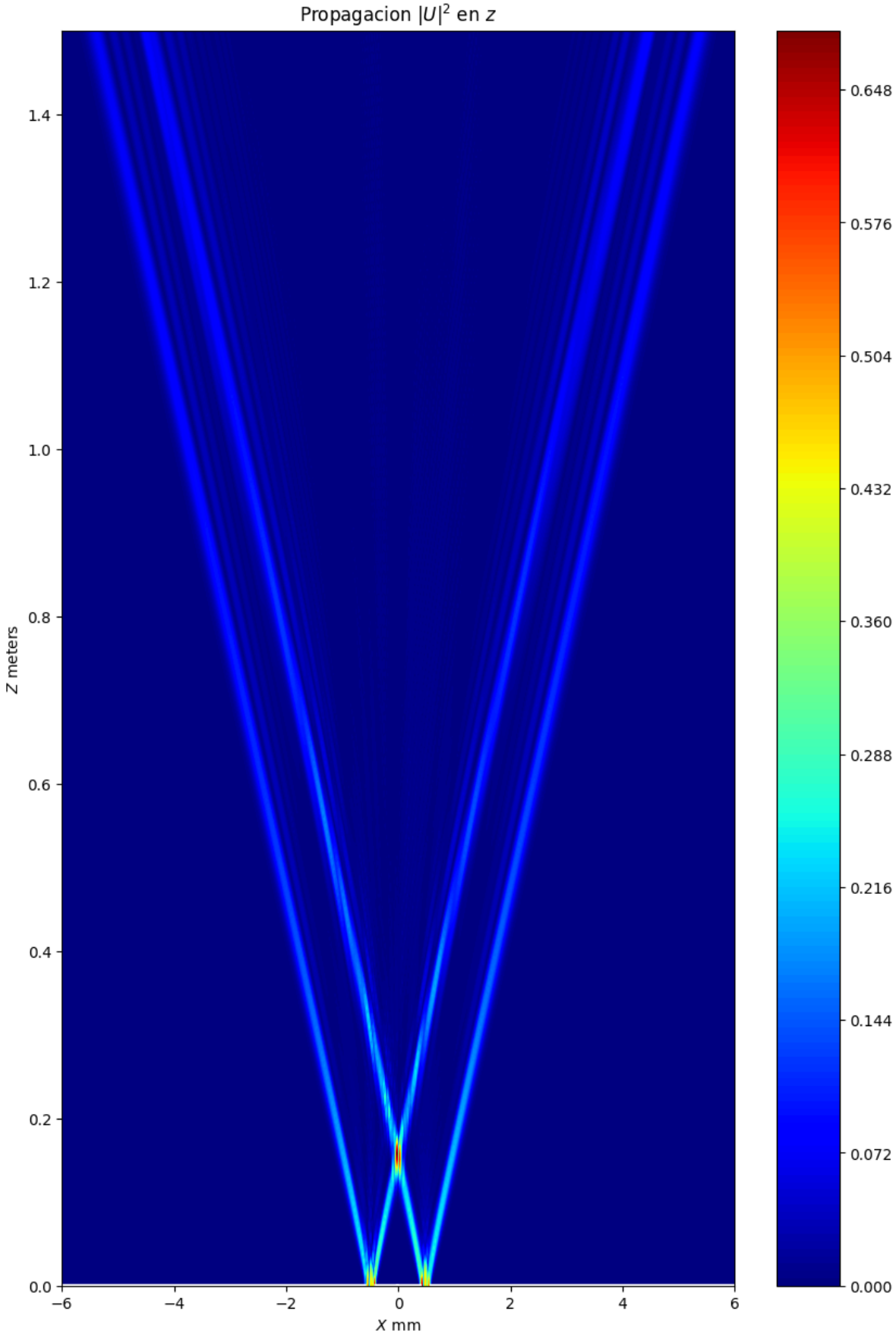
In [188…
```
po=np.exp(-((xs)**2)/0.000001)*np.sin(k*xs)


U[:,0]=0
U[:,-1]=0

U[0,:]=po
#Barrera izquiera
U[0,0:nxx-int((h/(2*dx)).round(0))-int((b/(2*dx)).round(0))]=0

#barrera derecha
U[0,nxx+int((h/(2*dx)).round(0))+int((b/(2*dx)).round(0)):-1]=0
U[0,-1]=0

# Barrera Central
U[0,nxx-int(h/(2*dx))+int(b/(2*dx)):nxx]=0
U[0,nxx:nxx+int((h/(2*dx)).round(0))-int((b/(2*dx)).round(0))]=0
```

In [188…
```
for i in range(0,nz-1):
    us=np.matmul(P,U[i,:])
    U[i+1,:]=us
```

In [188…
```
fig=plt.figure(figsize=(10,15))
ax = fig.add_subplot(111)
ax.clear()
prop=ax.contourf(X,Z,np.abs(U)**2,levels=200,cmap='jet')
plt.xlim([-6,6])
plt.colorbar(prop)
lwd=2.6
col='white'
#plt.xlim([-xf-2,xf+2])
ax.plot(xss[0:nxx-int(h/(2*dx))-int(b/(2*dx))],xss[0:nxx-int(h/(2*dx))-int(b/(2*dx)
ax.plot(xss[nxx+int(h/(2*dx))+int(b/(2*dx)):-1],xss[nxx+int(h/(2*dx))+int(b/(2*dx))
ax.plot(xss[nxx-int(h/(2*dx))+int(b/(2*dx)):nxx],xss[nxx-int(h/(2*dx))+int(b/(2*dx)
ax.plot(xss[nxx:nxx+int(h/(2*dx))-int(b/(2*dx))],xss[nxx:nxx+int(h/(2*dx))-int(b/(2
```

```
#plt.title('Propagacion abs(U)^2 en z')
plt.title(r'Propagacion $|U|^{2}$ en $z$', fontsize='large')
plt.xlabel(r'$X$ mm')
plt.ylabel(r'$Z$ meters')
```

Out[1888]: Text(0, 0.5, 'Z meters')



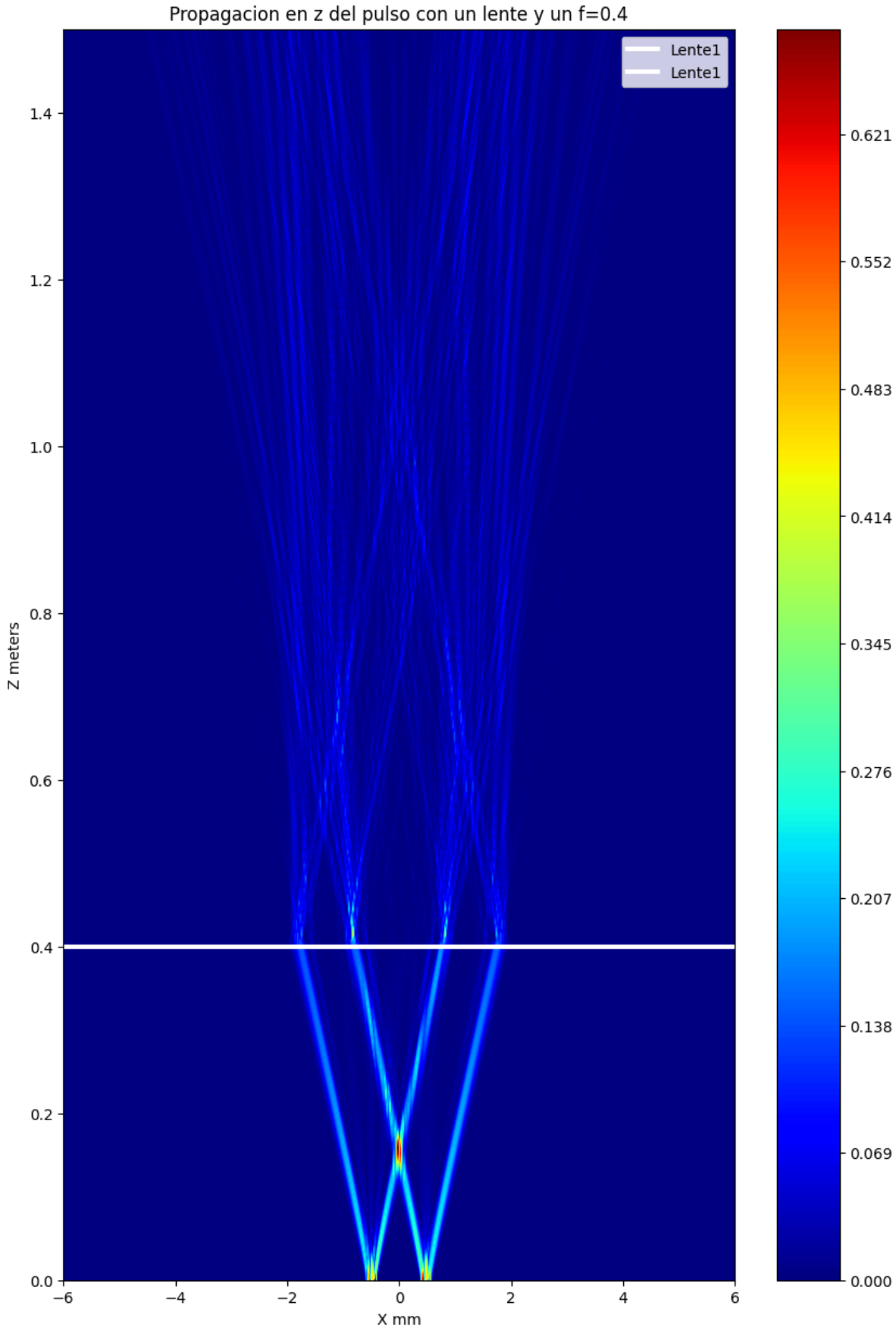
$$T_L(x) = \frac{f}{3} cos^3(kxf)$$

$$f = 0.4m$$

In [188…
```
f=0.4
Tl=(3)*np.cos((k*xs*f)**3)*f
#Tl=((-1j*k*xs**2)/(2*f))#**(1/6)
for i in range(0,nz-1):
    if zs[i].round(2)==f:
         us=np.matmul(P,U[i,:])*Tl
    else:
        us=np.matmul(P,U[i,:])
    U[i+1,:]=us
```

In [189…
```
fig3=plt.figure(figsize=(10,15))
ax = fig3.add_subplot(111)
ax.clear()
prop=ax.contourf(X,Z,np.abs(U)**2,levels=500,cmap='jet')
plt.colorbar(prop)

ff=np.ones(nx)*f
ax.plot(xss,ff,1,color='white',lw=3,label='Lente1')
plt.title('Propagacion en z del pulso con un lente y un f='+str(f))
plt.xlabel('X mm')
plt.ylabel('Z meters')
plt.legend()
```

Out[1890]: <matplotlib.legend.Legend at 0x1ac2c8a39a0>

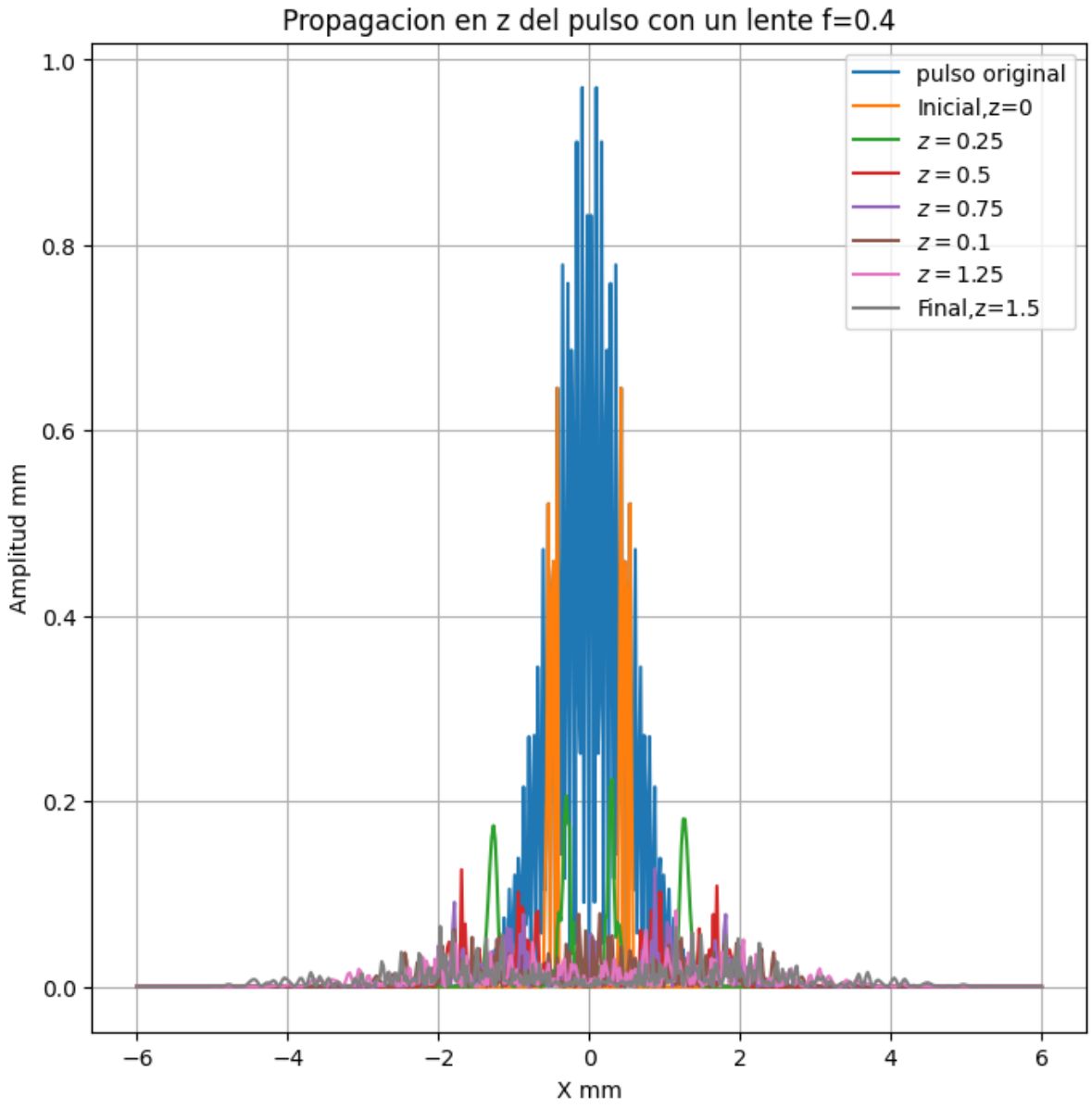


In [189…

```
fig5=plt.figure(figsize=(8,8))
plt.plot(xss,po**2,label='pulso original')
plt.plot(xss,np.abs(U[0,:])**2,label='Inicial,z=0')
```

```
plt.plot(xss,np.abs(U[int(nz/6)*1,:])**2,label='$z=0.25$')
plt.plot(xss,np.abs(U[int(nz/6)*2,:])**2,label='$z=0.5$')
plt.plot(xss,np.abs(U[int(nz/6)*3,:])**2,label='$z=0.75$')
plt.plot(xss,np.abs(U[int(nz/6)*4,:])**2,label='$z=0.1$')
plt.plot(xss,np.abs(U[int(nz/6)*5,:])**2,label='$z=1.25$')
plt.plot(xss,np.abs(U[-1,:])**2,label='Final,z=1.5')



#plt.axis('equal')
plt.grid('on')
plt.title('Propagacion en z del pulso con un lente f='+str(f))
plt.xlabel('X mm')
plt.ylabel('Amplitud mm')
plt.legend()
```

Out[1892]: <matplotlib.legend.Legend at 0x1ac2c897c70>



In [ ]: